www.vineyard-h2020.eu

VINEYARD

D2.1: Application requirements and specifications

DOCUMENT ID	D2.1
DUE DATE	31/07/2016
DELIVERY DATE	02/08/2016
CLASIFICATION	Public
AUTHOR/S	C. Strydis, G. Smaragdos, A. Bilas, C. Kachris, R. Jimenez, V. Spitadakis, C. Tsalidis, N. Chrysos
DOCUMENT VERSION	1.2

CONTRACT START DATE	1st FEBRUARY 2016
CONTRACT DURATION	36 Months

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission

Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

1 EXECUTIVE SUMMARY

Modern datacenter efficiency can be drastically improved through the inclusion of specialized hardware accelerators next to traditional CPU nodes. This approach has effectively given rise to heterogeneous datacenters. The main challenge, now, has shifted towards the taming of such heterogeneous ensembles of components so that the datacenter user and owner can take maximum benefit of the specialization offered. In the wake of such developments, the VINEYARD project aims at making hardware accelerators easy and transparent to use for significantly improving infrastructure efficiency while achieving application-side Quality of Service (QoS). VINEYARD proposes new approaches to integrating accelerators in datacenter environments and to dealing with heterogeneity and transparency issues. To evaluate the proposed solutions, VINEYARD starts from existing approaches and uses both application QoS and infrastructure efficiency as metrics. This deliverable starts with a basic taxonomy of datacenter applications in terms of QoS requirements. Based on this taxonomy, application-side requirements that are crucial for each of the considered VINEYARD use cases are introduced. Deliverable D2.1 has been extended to also include *infrastructure-side requirements*, such as resource utilization. Involving both types of requirements is an intriguing and challenging endeavor since many of these serve opposing objectives. The three VINEYARD use cases (neurocomputing, financial and data-management) have been extended by cloud applications. All involved applications are presented in detail and their respective requirements are given as design objectives for VINEYARD. Their requirements are summarized in the table below:

Application	**Requirements**		**Type**
	Throughput	**Latency**	
Neurocomputing	High to Very high	n/a	Batch processing
Financial	Moderate	Very low latency (msec)	On-line processing (Response-time-sensitive)
Transactional-analytics	High	Low latency (sec)	On-line processing (Rate-sensitive)

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission

Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

CONTRIBUTORS

Name	Organization
Christos Strydis	NEUR
Georgios Smaragdos	NEUR
Angelos Bilas	FORTH
Christoforos Kachris	ICCS
Ricardo Jimenez	LeanXcale
Vassilis Spitadakis	Neurocom
Christos Tsalidis	Neurocom
Nikolaos Chrysos	FORTH
Neil Morgan	STFC

PEER REVIEWERS

Name	Organization
Christoforos Kachris	ICCS
Nikolaos Chrysos	FORTH

REVISION HISTORY

Version	Date	Author/Organization	Modifications
0.1	16/04/2016	C. Strydis	Initial version
0.2	18/04/2016	A. Bilas	Sections, structure

0.3	18/04/2016	C. Kachris	Update on metrics, cost
0.4	25/04/2016	C. Strydis, G. Smaragdos	Neurocomputing application filled in
0.5	30/05/2016	C. Strydis	Modified structure, added material
0.6	30/05/2016	A. Almeida, C. Bravo	General structure and formatting
0.7	12/05/2016	C. Kachris, C. Tsalidis, R. Jimenez	New input for applications added
0.8	06/07/2016	C. Strydis	Modified structure, added material for NEUR
0.9	11/07/2016	C. Strydis, G. Smaragdos, A. Bilas	Modified structure, added material for NEUR
1.0	20/07/2016	C. Strydis, C. Tsalidis, C. Kachris, N. Chrysos	Incorporated material from all partners, reached document stable version
1.1	26/07/2016	C. Strydis, G. Smaragdos, N. Morgan	Added more material on applications, readjusted application-side requirements
1.2	02/08/2016	C. Strydis, V. Spitadakis, C. Kachris, N. Chrysos	Incorporated reviewer corrections, performed minor edits throughout

(Page intentionally blank)

Table of Contents

Table of Figures

Table of Tables

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

2 Introduction

VINEYARD aims at making accelerators easy and transparent to use for the purpose of significantly improving infrastructure efficiency while achieving application-side Quality of Service (QoS). VINEYARD proposes new approaches to integrating accelerators in datacenter environments and to dealing with heterogeneity and transparency issues. To evaluate the proposed solutions, VINEYARD starts from existing approaches and uses both **application QoS** and **infrastructure efficiency** as metrics.

This deliverable provides a basic *taxonomy* of applications in terms of QoS requirements. Based on this taxonomy, we then proceed to introduce the *application-side* requirements (and related metrics) that are crucial for each of the considered VINEYARD use cases. Then, we discuss *datacenter-side* – thus, infrastructure – requirements. Involving both types of requirements is an intriguing and challenging endeavor since many of these requirements serve opposing objectives (e.g. applications may strive for higher resource usage whereas datacenters may strive for lower resource usage).

2.1 Goal of deliverable

The primary goal of task T2.1 is to identify the requirements addressed by the participating applications of the use-cases, namely **neurocomputing**, **financial,** and **data-management** use-cases. Analysis of the various applications will include:

a. **Functional requirements:** Algorithms, functions, data flows will be defined and well specified.
b. **QoS (non-functional) requirements:** Acceptable latency, target throughput, target throughput per watt, (worst-case) execution time, scalability etc.

Infrastructure requirements posed by datacenter owners will also be presented and include:

a. **Resource utilization:** CPU, memory, accelerator, etc.
b. **Cost:** power consumption, maintenance etc.

2.2 Audience

This deliverable is intended for:

a. VINEYARD partners, including the application providers and infrastructure experts;
b. The research community that aims at addressing issues related to application QoS and infrastructure efficiency in modern datacenter servers; and
c. The industry related to the Cloud that is interested in understanding technological issues that affect cost and evolution of datacenter servers.

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

2.3 Document structure

Chapter 3 opens up with a general taxonomy of datacenter applications and, then, uses it to classify the VINEYARD applications. This classification is important firstly to help the reader better understand the field and, secondly, to easily identify which workload requirements are relevant to which workload class. The chapter, then, introduces application-side requirements which pave the way for the discussion of the various VINEYARD applications in the following Chapters 5, 6, 7 and 8.

Chapter 4 discusses infrastructure-side metrics that are related and affect application behavior. We include this discussion of infrastructure-side metrics because they affect overall application efficiency and cost.

Chapters 5, 6 and 7 present the functional and QoS (Quality-of-Service) requirements of the three VINEYARD application use-cases. A new application use-case (Cloud) has been included in Chapter 8 that studies widely-used cloud applications. This chapter has been added so as to study the requirements of cloud applications that are deployed in datacenters and to ensure that the VINEYARD framework can support a wide range of commonly-used applications. Application-side metrics are listed in these chapters, that is, metrics relevant from the datacenter-user perspective.

3 Datacenter workloads and requirements

In this chapter, we provide a general taxonomy of datacenter applications and, then, use it to classify the VINEYARD applications. This classification is important firstly to help the reader better understand the field and, secondly, to easily identify which workload requirements are relevant to which workload class. We, subsequently, introduce relevant application-side requirements and their expression in tangible metrics. This chapter will provide us with the toolset necessary to introduce the four different VINEYARD applications in the following Chapters 5, 6, 7 and 8.

3.1 Datacenter workload concepts and classification

Modern workloads tend to exhibit increased complexity in their structure, mainly due to the use of multiple components and the complexity of the underlying software stacks. Figure 1 shows such a stack that is used to provision resources and to execute different types of workloads.

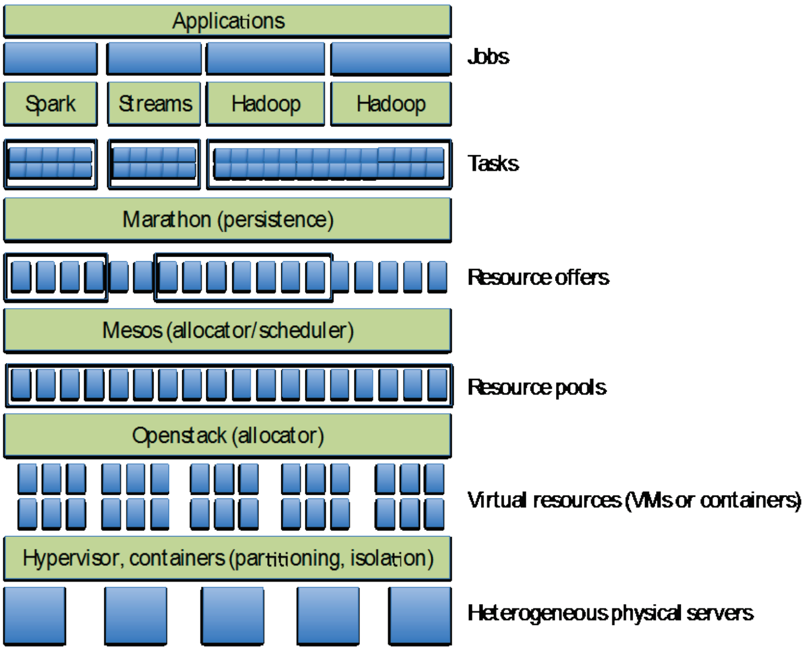

Applications
Jobs
Spark
Streams
Hadoop
Hadoop
Tasks
Marathon (persistence)
Resource offers
Mesos (allocator/scheduler)
Resource pools
Openstack (allocator)
Virtual resources (VMs or containers)
Hypervisor, containers (partitioning, isolation)
Heterogeneous physical servers

Figure 1 Typical software stack in a datacenter

In datacenter environments[1], a ***workload*** is a set of applications that run on a set of servers and make use of available resources. An application is a single- or multi-server (distributed) program that is typically run within a single administrative domain, e.g. by a single provider or client. Applications are started and have an expected execution time. Applications that run continuously, without terminating, are sometimes called services. Since this distinction does not affect the work in VINEYARD, we use the term applications for every program that executes on servers. Each application comprises one or multiple ***jobs***. A job is a collection of ***tasks*** with the same characteristics. Tasks are location-independent, they can run on any machine (or slice or Virtual Machine – VM), they will have access to their data over a global namespace, and they may communicate with other tasks, independently of location. Therefore, from the infrastructure point of view, we have tasks that originate in different jobs (applications and services) that collectively form the workload.

Jobs and tasks are typically divided into two broad categories, *batch- (or offline-) processing* and *online-processing* ones. Figure 2 shows these concepts.

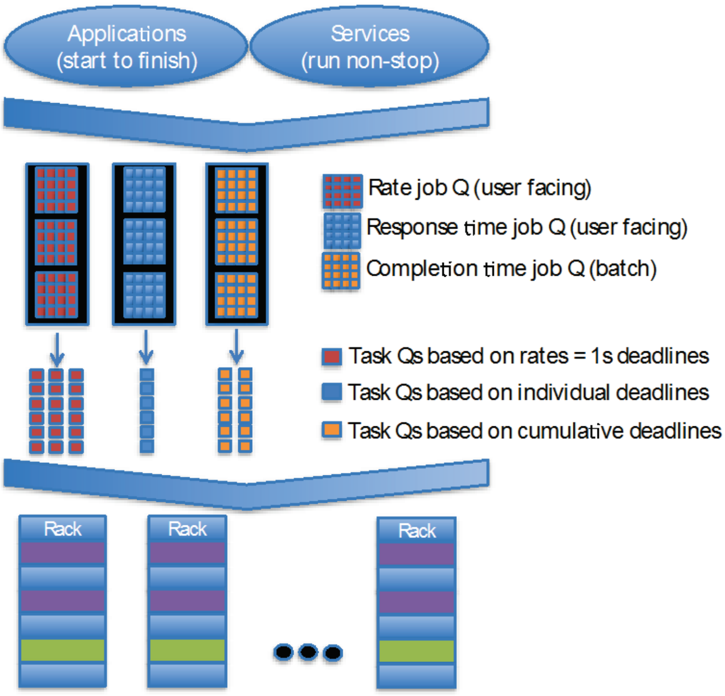

Applications
(start to finish)
Services
(run non-stop)
Rate job Q (user facing)
Response time job Q (user facing)
Completion time job Q (batch)
Task Qs based on rates = 1s deadlines
Task Qs based on individual deadlines
Task Qs based on cumulative deadlines
Rack
Rack
Rack

Figure 2 Applications, services, jobs, and tasks. Jobs and tasks may wait in queues (per requirement type) before execution

[1] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: flexible, scalable schedulers for large compute clusters. In Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys'13). ACM, New York, NY, USA, 351-364. DOI=http://dx.doi.org/10.1145/2465351.2465386.

Batch* (or *Offline*) *Processing: Data is processed in groups or batches. Batch processing is typically used for large amounts of data that must be processed on a routine schedule, such as paychecks or credit-card transactions. A batch-processing system typically collects, groups and processes transactions periodically. Batch programs require *no user involvement* and require significantly fewer network resources than online systems (discussed next).

Batch applications typically process high volumes of data that have been collected and stored. They usually involve complicated processing that needs to be performed on the data and they imply long execution times. The main performance metric for these applications is processing *throughput*. Batch jobs are typically required to finish by a relatively long (soft or hard) deadline, based on user-expectation or application requirements. Therefore, batch jobs can wait in a queue, so long as they do not violate their expected deadline.

Online Processing: An online system handles requests when they occur and provides output directly to users. Because it is interactive, online processing avoids delays and allows a constant dialog between the user and the system. The system processes requests completely when and where they occur. Users interact directly with the information system. Users can access data randomly. The information system must be available whenever necessary to support business functions. Therefore, online jobs and tasks have relatively short deadlines and they cannot generally afford to wait before starting execution because a user is waiting for the result. Online jobs and tasks, are sometimes called “user-facing” as well, since typically a user is waiting for the result.

Online-processing applications process high volumes of streaming data and usually the processing that needs to be done in these cases is simpler than in the case of batch-processing applications. Online jobs can be further broken down into two sub-categories:

1. **Jobs consisting of rate-sensitive tasks:** Tasks that need to complete at a certain rate;
2. **Jobs consisting of response-time-sensitive tasks:** Each task needs to complete within a deadline from the time it starts.

A conceptual illustration of batch and online jobs is shown in Figure 3, below.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

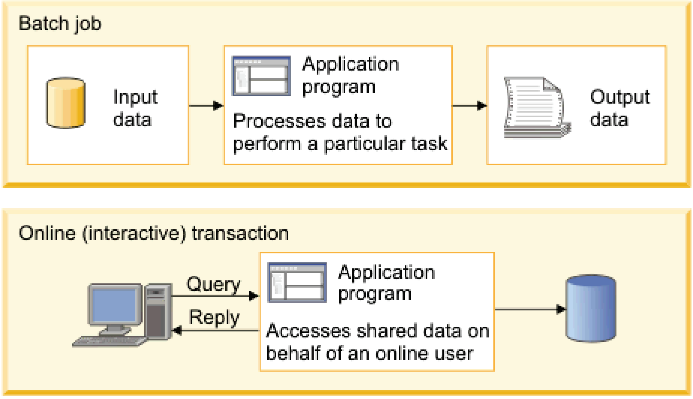

Batch job
Input data
Application program
Processes data to perform a particular task
Output data
Online (interactive) transaction
Query
Reply
Application program
Accesses shared data on behalf of an online user

Figure 3 Batch- (or Offline-) vs. Online-Processing applications in the datacenter[2]

3.2 Datacenter application requirements

A wide range of batch and online applications exists (see Figure 4). As the figure suggests, batch and online jobs (and tasks) are sensitive to different QoS metrics. Online jobs, that execute continuously, typically care either about task response time or task rate. Batch jobs, on the other hand, typically need to finish within the expected deadline and, therefore, care about job execution time without particular attention to tasks. Nevertheless, this rule is not absolute. For example, some online jobs have soft deadlines (e.g. a web search should return in a few seconds), and, in order to meet them, they may impose hard deadlines on the tasks they spawn[3].

Given that jobs are sets of tasks with similar characteristics, we can abstract this view of the world as follows. In modern infrastructures, we need to deal with multiple concurrent tasks, each with one of the following QoS metrics:

- Tasks sensitive to **response time** (latency); typically measured in milliseconds. Each task needs to finish by a specific deadline.
- Tasks sensitive to **rate** (or throughput); typically measured in tasks/second. Tasks need to maintain a completion rate within each measurement interval.
- Tasks sensitive to a **collective deadline**. All tasks need to finish by a cumulative deadline.

[2] Source: [Online Available:] https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_mfworklds.htm

[3] Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74-80.

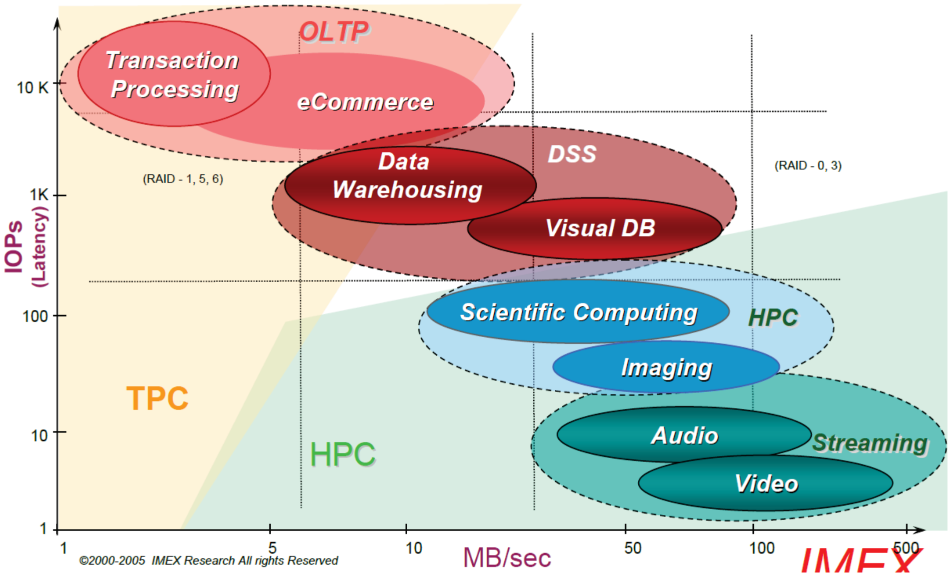


Figure 4 Various batch (high-throughput) and online datacenter applications[4]

Additionally, users or providers might care to optimize either for *average* or *tail* statistics. Average statistics can be calculated over all or a portion of the tasks. For instance, a provider might care about minimizing the average response time of 100% or 95% of the requests. Tail statistics indicate the percentage of requests that are within an acceptable QoS target. For instance, a provider might care to optimize infrastructure such that 99.99% of user-facing tasks are within 2x of the average response time. As another example, a user-facing job may care that 99% of its tasks finish within 15 microseconds. Typically, today, we assume that tasks arrive labeled with the QoS category they belong to (response time, rate, average or tail, soft or hard deadline and the associated targets) and it is up to the system to optimize for the corresponding QoS metrics.

Nowadays, there is general belief that modern infrastructure can achieve the required performance statistics, especially when over-provisioned. On the other hand, it has become of paramount importance to achieve the performance at high infrastructure utilization (see Section 4.2 for an in-depth discussion). Additionally, tail statistics are particularly important today for datacenter workloads because jobs issued by users, e.g. a user-query, are broken down to multiple tasks and are sent to different servers, sort of in a fan-out scheme. If – let's assume – each user request is broken down to 100 tasks and 99% of the tasks are close to the average response time but 1% is far from it, then each user query will have a "slow" task. Eventually, 63% of **all** user queries will be slow[5].

[4] [Online Available:] http://www.slideshare.net/IMEXresearch/next-generation-data-centers-11750006

[5] Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74-80.

Based on the above, the following list of application requirement types (and related metrics) are being considered in VINEYARD (see Table 1). In the chapters that follow, each application provider will describe their respective applications and list those requirements pertinent for their specific cases. In VINEYARD, we aim to capture requirements based (a) on a common set of metrics typically used in characterizing datacenter-deployed applications as well as (b) on a potential set of extra, application-specific requirements, if desired by the application providers.

Table 1 VINEYARD list of application requirements. Can refer to tasks, jobs and whole applications.

Requirement	Definition	Related metric (in units)
Task response time (or latency) **(T_t)**	Turnaround or response time (average, tail, total) of a task; that is, the time required between providing new input to the datacenter and getting corresponding results back. *Pertinent to online applications.*	Milli-seconds
Task rate (or throughput) **(R_t)**	Number of tasks that must be completed per second. *Pertinent to batch applications.*	Units/sec (Units can be tasks, transactions, FP operations or some other app-related quantity)
Job execution time **(T_j)**	The time from submission (when a job is available to start) until finish (when results are available). In this definition, execution time includes queueing delays as well.	Seconds, minutes, or hours

3.2.1 Programming effort as a special application requirement

Except for the above requirements, a potentially interesting requirement that would be interesting to capture is the ***programming effort***, that is, the effort spent to deploying an application to a datacenter setup. The reason such a requirement could be interesting and relevant for the application provider (i.e. the datacenter user) is that it would directly affect the amount of time it takes to port e.g. a reference C-code application to a typical datacenter setup based on CPUs. In the case of VINEYARD and that of future aspired datacenters, this programming effort becomes even more crucial as **it could potentially capture the additional effort needed to port (or re-port) an application to a heterogeneous, accelerator-based datacenter**. It is important to note that programming effort can be directly affected by the amount of optimizations performed. Such is the case – for instance – of performing vectorization optimizations in a generic Xeon-Phi platform.

Estimating or clocking such programming costs might capture part of the trade-off between porting a simpler version of an application to a standard, CPU-based datacenter with limited performance (efficiency) gains versus porting a more complicated version to an accelerator-based datacenter with significantly improved gains, but at a potentially recurring extra time cost. Last but not least, the reader can imagine that such a requirement may also be of interest to the datacenter provider (or other third party) as well, since it is often the case that application providers outsource the code-deployment effort of their own applications to the datacenter providers (or other stakeholders of the datacenter ecosystem) due to lack of coding expertise. In such cases, electing to dedicate more time in mapping (part of) an application to a particular datacenter accelerator may cost extra programming time but may yield significant savings on the part of the datacenter in the form of denser performance per watt or dollar.

The main reason that *programming effort* has not been added to the primary list of application (or datacenter requirements) is the difficulty that lies in assessing it in the form of a metric. Various metrics are being considered ranging from standard Lines-of-Code (LoC) and algorithmic complexity (e.g. big-O notation) of designed programs, to application-specific metrics like employee person-months/average employee salary or some such custom metric. Due to its potential significance, we leave this requirement as an optional one and shall revisit it during the evaluation phase of the project, provided that sufficiently robust information on programming effort has been collected across the various VINEYARD applications and with respect to reference (CPU-only) as well as improved (accelerator-based) implementations.

3.3 Overview of VINEYARD applications

There are 7 applications considered in VINEYARD across 3 domains, as follows:

- two **financial** applications (Pre-Trade Real-Time Risk-Management System and Market-Surveillance Real-Time System);
- one class of **neurocomputing** applications (high-performance Inferior-Olive simulation);
- four **data-management** benchmarks representing enterprise workloads (TPC-C, TPC-H, YCSB, LinearRoad); and
- **Cloud applications** based on Apache Spark.

Based on our established taxonomy, the financial and the transactional VINEYARD applications are *online* applications, while the neurocomputing application is a *batch* application. In terms of the cloud applications, we use the SparkBench[6] Benchmark that includes both batch and online applications.

The following figure depicts a high level overview of the applications requirements in terms of throughput and latency. The specific application requirements of each application will be described in detail in the following chapters. This figure shows that each use-case has different characteristics and that the selected applications cover a wide range (classes) of data center applications.

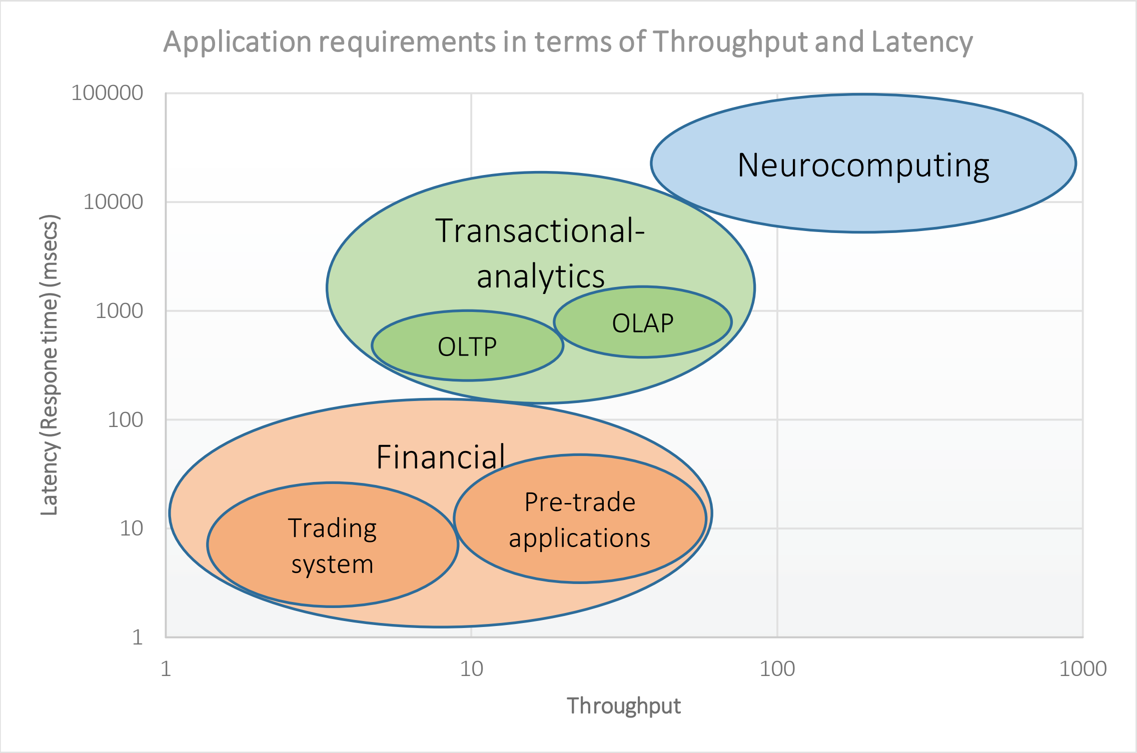


Figure 5 Application requirements of the VINEYARD use-cases

[6] Min Li, Jian Tan, Yandong Wang, Li Zhang, and Valentina Salapura. 2015. SparkBench: a comprehensive benchmarking suite for in memory data analytic platform Spark. In Proceedings of the 12th ACM International Conference on Computing Frontiers (CF '15). ACM

4 Infrastructure-side requirements

Challenges in datacenters are driven by conflicts between application and infrastructure trends. For this reason, in this chapter we present infrastructure-side (i.e. datacenter) requirements. In addition, we introduce and discuss the interesting concept of *combined metrics* (i.e. pertaining to both application and infrastructure).

4.1 Application-side vs. infrastructure-side requirements

Nowadays, applications in datacenters (and the Cloud) have challenging QoS requirements. For instance, services that respond to user requests over the web, typically need to complete each request in a few milliseconds. Although traditionally new infrastructures and technologies have targeted to improve average statistics (latency or response time), today this is not sufficient. Instead, it is important to optimize QoS not in the average case, but for all requests. As an example, consider the case where a user request breaks down to a 100 smaller requests that form a single web page that the user will view. If 1 every 100 requests is slow, then every user request will be slow.

Given such requirements, providers tend to over-provision infrastructure, to avoid spikes in application QoS degradation. This results in low server (infrastructure) utilization. In fact, application and infrastructure metrics form an important trend: *Allow for unpredictable application QoS degradation or keep infrastructure utilization low.* A main challenge is to find better solutions to this tradeoff than what is possible today. To achieve this, it becomes important to not only consider application- but infrastructure-side metrics, as well.

4.2 Infrastructure-side requirements

4.2.1 Infrastructure software stack

Modern data centers tend to use complex software stacks (Figure 1) to satisfy job and task requirements over shared physical resources. This stack is necessary in order to ensure that multiple clients can have access to the same set of shared resources. Applications and services arrive at the datacenter from clients. In most cases, an underlying layer, such as Spark or Hadoop, (aka application runtime framework) typically partitions client requests into computation chunks (tasks) that can take advantage of the datacenter's multiple available computing units. A layer below the runtime, such as Marathon, ensures the persistence of resource allocation to applications and services to avoid frequent resource allocation requests. At this point, the resources that the applications can use may be physical and/or virtual. A further layer (such as Mesos) is used to perform resource allocation from a pool of resources offered by yet another layer, such as OpenStack. Finally, in case of virtualized resources, a hypervisor or a container management system is necessary to partition physical resources into virtual chunks.

4.2.2 Infrastructure-side metrics

Nowadays, with increasing energy and maintenance costs, infrastructure providers strive to improve the efficiency of their infrastructure, because this is directly related to the *total cost of ownership (TCO)* and the *return on investment (ROI)*. The goal is to use infrastructure *as much as possible* during its lifetime. TCO is typically broken down to two components:

- **Capital expenses (CapEx):** The cost of acquiring equipment.
- **Operational expenses (OpEx):** The cost of maintaining equipment over its lifetime.

Today, both of these costs are substantial and constitute significant concerns. However, in the long term, it is projected that consumed energy and dissipated power (typically part of OpEx) will dominate TCO. Therefore, research today aims at optimizing power and energy of modern infrastructures.

Although **cumulative (coarse-grain) energy and power** are measured and monitored directly today for full servers or racks, it is more challenging to obtain finer grain measurements for individual components, such as processors, memories, storage devices, network devices, controllers, and accelerators.

A first approximation for energy and power can be obtained by examining the utilization of the different components. Generally, a component consumes more energy and dissipates more power at higher utilization. However, it is important to note that components and systems do not exhibit "energy proportionality"[7]. For instance, a server may dissipate 75% of its nominal power at idle state. Therefore, the goal is to keep infrastructure components as utilized as possible during their lifetime. Previous work has looked into models that estimate power and energy based on component utilization[8] and has proposed various models for estimating power when actual power and energy measurements are not available.

In VINEYARD, we will measure both utilization and power of resources, at different granularity and we will augment our measurements with appropriate models. Overall, our goal is to capture the notion of efficiency of different (significant) components, such as CPUs and accelerators in terms of utilization, energy, and power.

[7] Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer* 40, 12 (December 2007), 33-37. DOI=http://dx.doi.org/10.1109/MC.2007.443.

[8] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture* (ISCA'07). ACM, New York, NY, USA, 13-23. DOI=http://dx.doi.org/10.1145/1250662.1250665.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

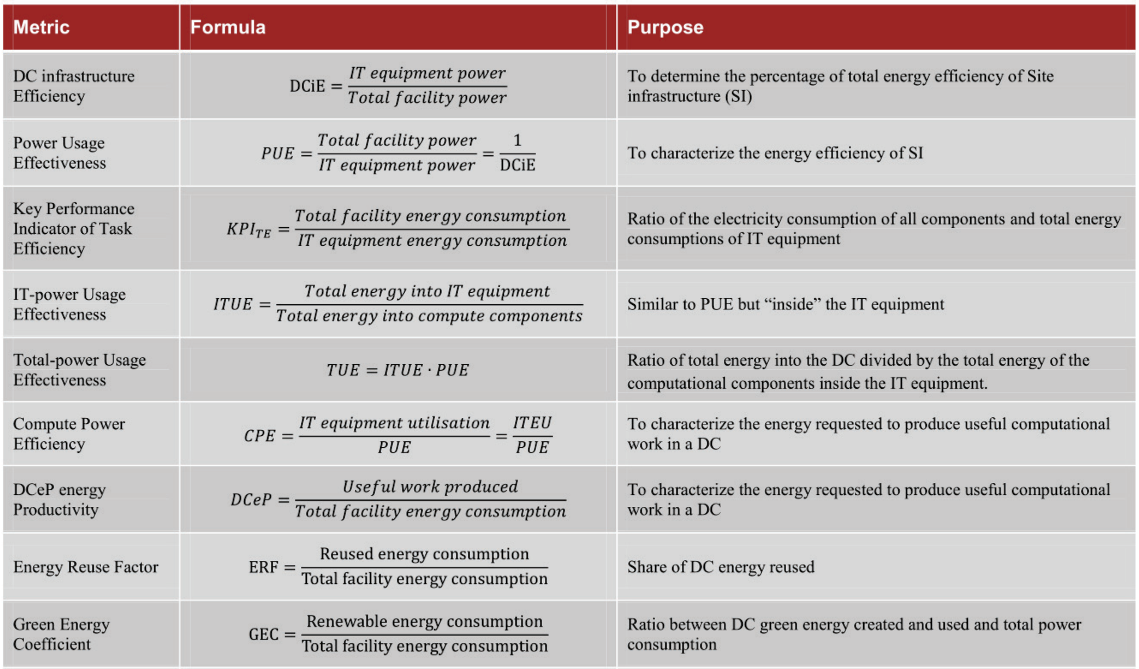
Table 2 VINEYARD list of infrastructure requirements. Can refer to individual tasks or whole jobs.

Requirement	Definition	Related metric (in units)
Energy consumption	Energy cost (average, peak) for executing a single task. *Pertinent to batch applications.*	Joules
Power dissipation	Power cost (average, peak) for executing a single task. *Pertinent to online applications.*	Watts
Power efficiency **(compound req.)**	Rate at which useful computations are effected per a given power budget. Can be calculated based on the aforementioned performance and power metrics. *Pertinent to online applications.*	IPS/W, FLOPS/W, transactions/W, etc.
Energy efficiency **(compound req.)**	Rate at which useful computations are effected per a given energy budget. Can be calculated based on the aforementioned performance and energy metrics. *Pertinent to batch applications.*	IPS/J, FLOPS/J, transactions/J etc.
Cost efficiency (compound req.)	Rate at which useful computations are effected within a certain expenditure threshold. Can be *partially* calculated based on the aforementioned performance and power/energy metrics. *Pertinent to both batch and online applications.*	IPS/$, FLOPS/$, transactions/$, etc.

More generally speaking, there are various architectural metrics that are used to measure the performance of the datacenters. The following table (Table 3) lists some of the most common data-center metrics used for energy efficiency[9].

[9] D7.1 Description of the energy metrics for data centres, DC4Cities, Project Nº 609304, 2014

Table 3 Most widely used datacenter metrics for energy efficiency



Metric	Formula	Purpose
DC infrastructure Efficiency	$\mathrm{DCiE} = \frac{IT\ equipment\ power}{Total\ facility\ power}$	To determine the percentage of total energy efficiency of Site infrastructure (SI)
Power Usage Effectiveness	$PUE = \frac{Total\ facility\ power}{IT\ equipment\ power} = \frac{1}{\mathrm{DCiE}}$	To characterize the energy efficiency of SI
Key Performance Indicator of Task Efficiency	$KPI_{TE} = \frac{Total\ facility\ energy\ consumption}{IT\ equipment\ energy\ consumption}$	Ratio of the electricity consumption of all components and total energy consumptions of IT equipment
IT-power Usage Effectiveness	$ITUE = \frac{Total\ energy\ into\ IT\ equipment}{Total\ energy\ into\ compute\ components}$	Similar to PUE but “inside” the IT equipment
Total-power Usage Effectiveness	$TUE = ITUE \cdot PUE$	Ratio of total energy into the DC divided by the total energy of the computational components inside the IT equipment.
Compute Power Efficiency	$CPE = \frac{IT\ equipment\ utilisation}{PUE} = \frac{ITEU}{PUE}$	To characterize the energy requested to produce useful computational work in a DC
DCeP energy Productivity	$DCeP = \frac{Useful\ work\ produced}{Total\ facility\ energy\ consumption}$	To characterize the energy requested to produce useful computational work in a DC
Energy Reuse Factor	$\mathrm{ERF} = \frac{\text{Reused energy consumption}}{\text{Total facility energy consumption}}$	Share of DC energy reused
Green Energy Coefficient	$\mathrm{GEC} = \frac{\text{Renewable energy consumption}}{\text{Total facility energy consumption}}$	Ratio between DC green energy created and used and total power consumption

The Power-Usage Effectiveness (PUE) and the IT Power-Usage Effectiveness (ITUE) are the most widely used metrics for the evaluation of the energy efficiency of the data centers. In VINEYARD, we aim to reduce significantly the energy consumption (and thus increase the Power Usage Effectiveness) by the efficient deployment of hardware accelerators that can complete specific tasks offloaded from the processors using much less energy than the general-purpose processors.

Besides the metrics for capturing power consumption in the datacenters, there are also some commonly used metrics specifically for the IT infrastructure in the datacenters. Table 4 depicts metrics for the energy efficiency of the IT infrastructure in the data centers. In VINEYARD, we aim to increase the IT productivity per Embedded Watt (IT-PEW), the IT Equipment efficiency and the Space, Watt and Performance efficiency (SWaP) metrics.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen’s University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission

Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Table 4 IT power metrics on datacenters

Metric	Formula	Purpose
IT Productivity per Embedded Watt (IT-PEW)	$\mathrm{IT-PEW} = \frac{\text{IT work}}{\text{IT energy use}}$	Measures IT energy productivity, work defined as network transactions storage or computing cycles
IT energy Productivity (ITeP)	$\mathrm{ITeP} = \frac{\text{IT useful work}}{\text{IT energy use}}$	Computes IT productivity as sum of weighted use of completed tasks relative to IT energy use
IT Asset Efficiency (ITAE)	$\mathrm{ITAE} = \mathrm{ITEE} \cdot \mathrm{ITEU}$	Computes energy productivity and resource efficiency of IT systems
IT Equipment Efficiency (ITEE)	$\mathrm{ITEE} = \frac{\text{IT work [OPS/IOPS/GbPS]}}{\text{IT energy use}}$	Computes IT equipment energy efficiency
IT Equipment Utilization (ITEU)	$\mathrm{ITEU} = \frac{\text{IT capacity used}}{\text{IT capacity installed}}$	Measures IT equipment utilization
Data Center & Server compute Efficiency (DCcE/ScE)	$\mathrm{DCcE} = \sum_{i=1}^{n} \mathrm{ScE}_i = \sum_{i=1}^{n} \frac{\text{primary services server i}}{\text{total services server i}}$	Measures share of primary services relative to secondary IT services (virtualization, backup, etc)
DC storage Efficiency (DCsE) / DC network Efficiency (DCnE)	For the efficiency of storage system / For the efficiency of network system	Will assign for the efficiency of storage system/ network system
Space, Watts and Performance (SWaP)	$\mathrm{SWaP} = \frac{\text{Performance}}{\text{SpacexPowerConsumption}}$	Gives an comprehensive assessment of the server's efficiency

4.2.2.1 Requirements concerning datacenter vendors

The following table (Table 5) shows datacenter requirements from the viewpoint of technology scaling, based on the ITRS 2015 Report. As is clear from the table, while the number of cores and the main memory will keep increasing significantly in the future, the total power consumption per server unit will remain almost the same (~700W), due to thermal constraints. Therefore, more energy-efficient solutions need to be adopted in order to keep the overall power consumption low.

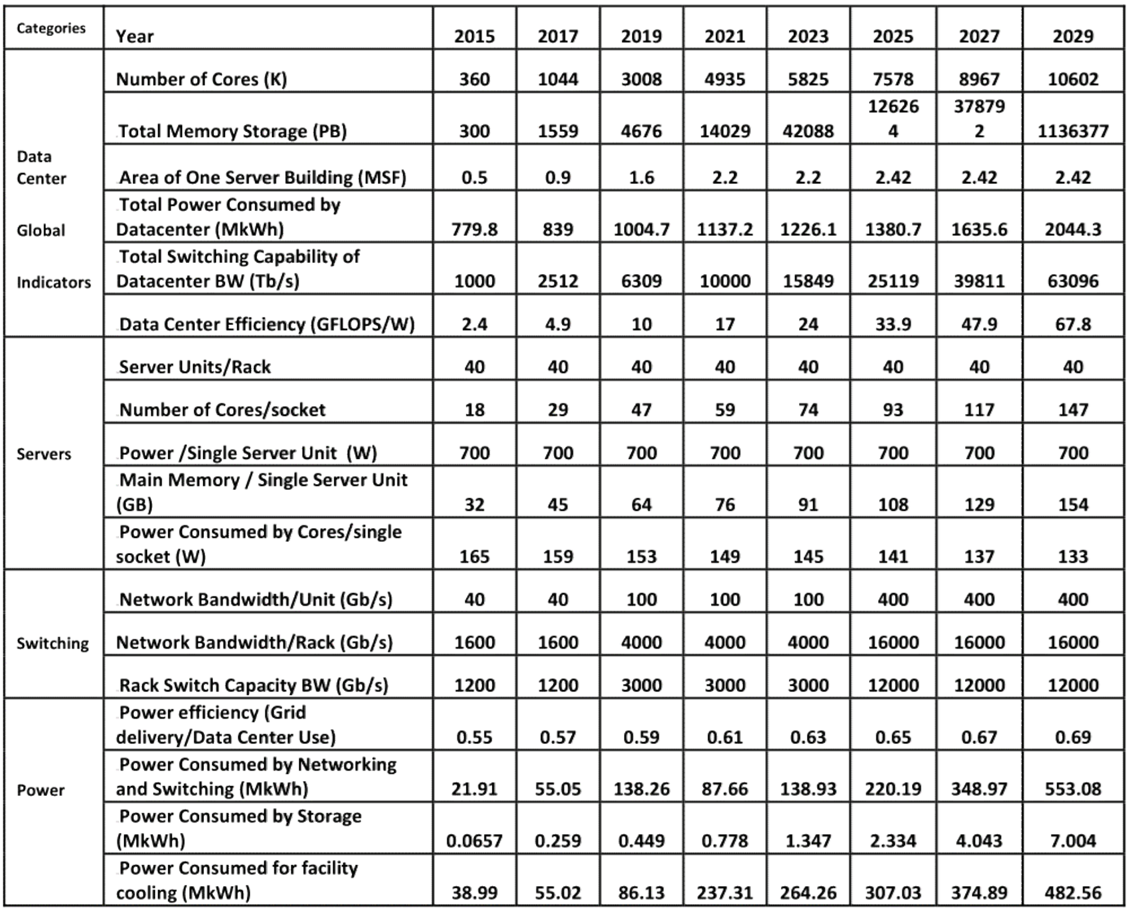
As is also shown in the table, datacenter efficiency in terms of GFLOPS/W will have to increase from 2.4 GFLOPS/W to 10 GFLOPS/W by 2019 and to 24 GFLOPS/W by 2023. Therefore, novel solutions such as the use of hardware accelerators that are more energy-efficient need to be adopted. ITRS has recently identified hardware accelerators as one of the key elements for the reduction of the power consumption in the sector of BigData and cloud computing (see Figure 6).

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

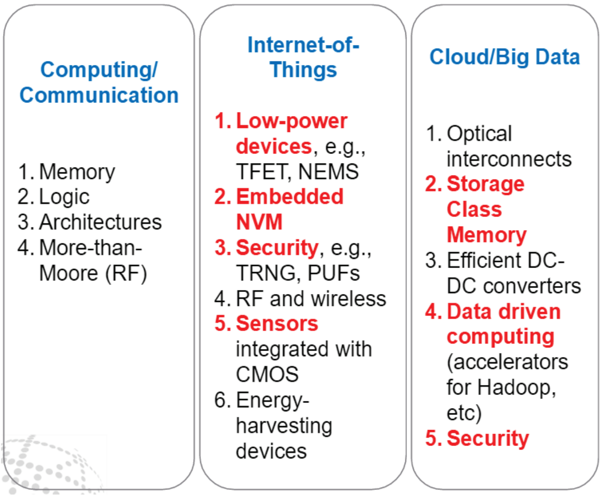
European Commission

Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Table 5 Datacenter requirements projections [Source: ITRS 2015, Executive summary]



Categories	Year	2015	2017	2019	2021	2023	2025	2027	2029
Data Center Global Indicators	Number of Cores (K)	360	1044	3008	4935	5825	7578	8967	10602
	Total Memory Storage (PB)	300	1559	4676	14029	42088	126264	378792	1136377
	Area of One Server Building (MSF)	0.5	0.9	1.6	2.2	2.2	2.42	2.42	2.42
	Total Power Consumed by Datacenter (MkWh)	779.8	839	1004.7	1137.2	1226.1	1380.7	1635.6	2044.3
	Total Switching Capability of Datacenter BW (Tb/s)	1000	2512	6309	10000	15849	25119	39811	63096
	Data Center Efficiency (GFLOPS/W)	2.4	4.9	10	17	24	33.9	47.9	67.8
Servers	Server Units/Rack	40	40	40	40	40	40	40	40
	Number of Cores/socket	18	29	47	59	74	93	117	147
	Power /Single Server Unit (W)	700	700	700	700	700	700	700	700
	Main Memory / Single Server Unit (GB)	32	45	64	76	91	108	129	154
	Power Consumed by Cores/single socket (W)	165	159	153	149	145	141	137	133
Switching	Network Bandwidth/Unit (Gb/s)	40	40	100	100	100	400	400	400
	Network Bandwidth/Rack (Gb/s)	1600	1600	4000	4000	4000	16000	16000	16000
	Rack Switch Capacity BW (Gb/s)	1200	1200	3000	3000	3000	12000	12000	12000
Power	Power efficiency (Grid delivery/Data Center Use)	0.55	0.57	0.59	0.61	0.63	0.65	0.67	0.69
	Power Consumed by Networking and Switching (MkWh)	21.91	55.05	138.26	87.66	138.93	220.19	348.97	553.08
	Power Consumed by Storage (MkWh)	0.0657	0.259	0.449	0.778	1.347	2.334	4.043	7.004
	Power Consumed for facility cooling (MkWh)	38.99	55.02	86.13	237.31	264.26	307.03	374.89	482.56



Computing/ Communication

1. Memory
2. Logic
3. Architectures
4. More-than-Moore (RF)

Internet-of-Things

1. **Low-power devices**, e.g., TFET, NEMS
2. **Embedded NVM**
3. **Security**, e.g., TRNG, PUFs
4. RF and wireless
5. **Sensors** integrated with CMOS
6. Energy-harvesting devices

Cloud/Big Data

1. Optical interconnects
2. **Storage Class Memory**
3. Efficient DC-DC converters
4. **Data driven computing** (accelerators for Hadoop, etc)
5. **Security**

Figure 6 Hardware accelerators have been identified as a potential emerging domain for the reduction of power consumption in the datacenters [Source: ITRS Roadmap 2015, Beyond-CMOS Technology Roadmap]

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

4.2.2.2 Requirements concerning datacenter operators

Understanding the energy consumption associated with all elements of a datacenter is essential to accurately define the total cost of ownership and drive efficiency savings at each layer of the infrastructure stack. The ability to accurately monitor infrastructure and environmental conditions within the datacenter is essential. Providing the ability to optimize available resources and reduce costs as well as providing valuable data to inform strategic planning. The ability to combine system and infrastructure data with application run data enables intelligent scheduling of workloads to increase throughput while minimizing costs.

The ability to query systems and supporting infrastructure via integrated protocols such as Modbus, BACnet, SNMP and IMPI provides data on power and energy usage as well as diagnostic monitoring information. The use of building-management systems (BMS) or other Data-Center Integration-Management (DCIM) tools allows automated monitoring and reporting. Together with environmental data such as temperature and humidity and workload data (power, energy and runtime) it is possible to analyze the impact of given workloads across the infrastructure stack. This data can be used to accurately estimate power and energy consumption and inform job scheduling policies. Given variations in energy cost during a 24 hour cycle it is common to schedule jobs to run given times of the day or night maximize throughput whilst minimizing energy consumption.

Predicting the energy and performance profile characteristics of applications on a range of hardware can also be used as part of an energy aware scheduling approach. When considering system level energy consumption it is valuable to understand both the dynamic and static energy consumption of the hardware. For Intel based machines with the Running Average Power Limiting (RAPL) facility it is possible to employ Dynamic Voltage Frequency Scaling (DVFS) to potentially reduce the overall dynamic energy consumption of an application by throttling the CPU frequency. This technique is not applicable to all applications but has been shown for some application to save overall energy whilst incurring a modest increase in overall runtime. The RAPL feature can also be used to suspend systems or individual nodes within a system and place them in a low power state sleep state when not in use. Reducing the static energy consumption of systems that have low or sporadic utilization this can lead to significant energy savings.

Together these techniques can be used to correlate power and energy data along with application level monitoring and prediction to validate the accuracy of the power and energy consumption.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Requirement	Definition	Related Metric
Infrastructure and Systems Power and Health Status Monitoring via BMS and/or DCIM solutions. e.g. Chillers, Cooling Loop Pumps, Cooling Towers, Transformers, Universal Power Supplies (UPS).	Ability to monitor power consumption and infrastructure health data / performance metrics via SNMP, BACnet and Modbus. E.g. Cooling Tower Efficiency, Transformer and UPS measurement of loss.	Power (Watts) Temperature (°C / °F) Efficiency % of optimal.
Data Centre Environmental Monitoring.	Measurement and recording of temperature and humidity within the data centre to identify hotspots and potentially hazardous increases in humidity.	Humidity (%) Temperature (°C / °F)
Application Performance Monitoring via scheduling software or other third party tools e.g. IPMI.	Monitor application level power and energy consumption. Correlate, job, resource and licence data across all systems.	Runtime (Seconds) Power (Watts) Energy (Joules)

4.2.2.3 Requirements concerning datacenter users

The VINEYARD application providers have attempted to propose various datacenter-side requirements that – in the context of their respective applications – may improve datacenter performance, and can – indirectly – also lead to improvements on their deployed datacenter applications. Such requirements from the standpoint of application providers can be found in the respective application chapters, in Sections 5.3, 6.3 and 7.3.

4.3 Combined metrics

In the previous sections we have discussed significant metrics used today from the application and infrastructures sides. Today, it is fairly straight-forward to optimize for either of these metrics individually.

From the application side, a provider can target a low degree of consolidation and run an application on an over-provisioned server. This will generally result in good application behavior. This over-provisioning approach is used today because in many market segments it is preferable to keep customers happy rather than be as efficient as possible.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Amazon reports that it loses 1% of sales for an increase of 100 ms in response latency. Also, this approach has historically been fueled by improvements in technology that allow us to create larger servers with more and more resources (cores, memory, network, storage).

On the other hand, if we consider only the infrastructure-side metrics, it is relatively easy to drive utilization high, by consolidating many applications on a single server, typically via virtualization or containers. This has been a trend over the last few years, as there are increasing (cost) concerns about running servers at low utilization.

The main challenge we face today is to achieve a balance between the two sides: To operate infrastructure at high utilization without damaging user experience.

To help us evaluate alternatives and solutions in this direction, we plan to use combined metrics that take into account both application QoS and infrastructure efficiency. Previous work has taken such steps, however, there are a number of challenges to address. Such combined metrics are generally not used today. Our goal in VINEYARD is to refine existing approaches and introduce new ones for evaluating the use of accelerators in future datacenter infrastructures.

5 Neurocomputing application

5.1 Application description & functional requirements

The neurocomputing application of VINEYARD is a highly accurate computational model of the inferior-olivary nucleus in the brain. Each biological neuron in that nucleus (i.e. biological neuronal network) is an electrochemically excitable cell that processes and transmits signals within the brain. The biological neuron comprises, generally speaking, three parts (called *compartments* in neuromodeling jargon): The Dendrites, the Soma and the Axon. The dendritic compartment represents the cell input stage. The dendrites pick up electrochemical stimuli from other cells and transfer them to the soma. In turn, the soma processes the stimuli and translates them into a cell membrane potential, which evokes a cell response called an action potential or, simply, a spike[10]. This response is transferred through the axon, which is the cell output stage, to other cells. An electrochemical connection between two cells (axon-dendrite) is called a synapse. This is a simplified description of the neuron. In reality, electrochemical processes are taking place in every neuron compartment. Potentially a neuron model can have hundreds of compartments.

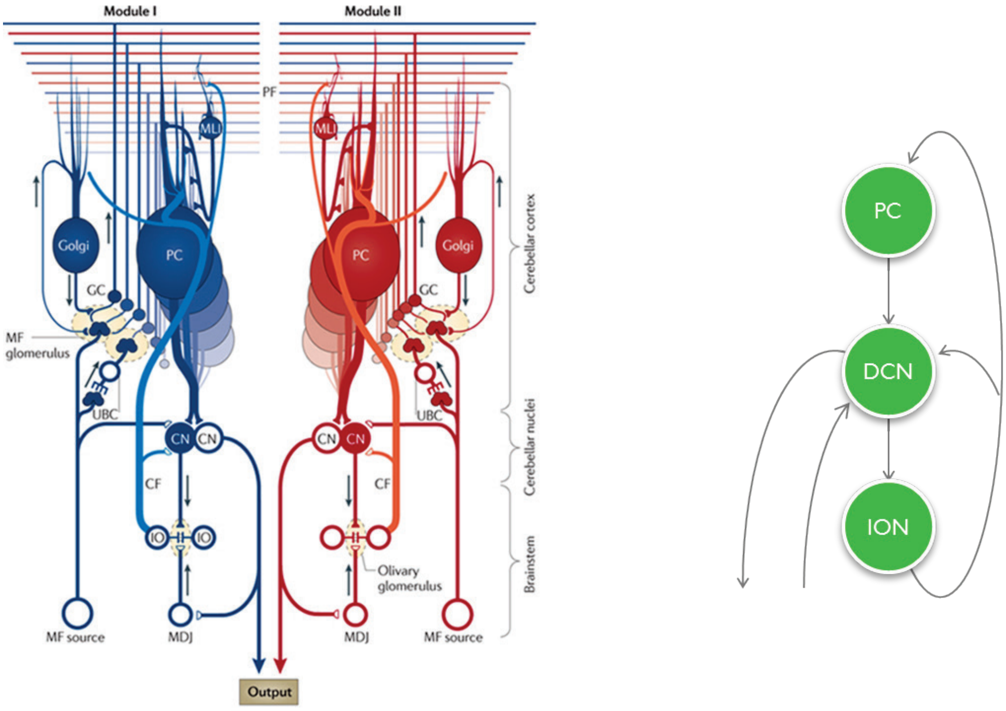


Figure 7 The olivocerebellar system and its basic modeling abstraction

[10] G. Wulfram and W. Werner. Spiking Neuron Models. Cambridge University Press, 2002.

The inferior-olive application (abbrev. *InfOli*) is a model that represents the inferior-olivary nucleus (ION). This is an intricate part of the olivocerebellar system which is one of the most dense brain regions and plays an important role in sensorimotor control. It does not initiate movement by itself but it does provide rhythm and coordination for motor functions. It is considered to be imperative for the instinctive learning and smooth completion of motor actions. The inferior olive provides one of the two main inputs to the olivocerebellar system through the so-called *climbing fibers*, the other being the *mossy fibers*. The inferior-olivary neurons are also heavily interconnected to one another through direct *electrical* connections between their dendrites, called *gap junctions* (GJs). Gap junctions define the synchronization behavior between the inferior-olivary cells and, subsequently, influence the synchronization and learning properties of the overall system[11].

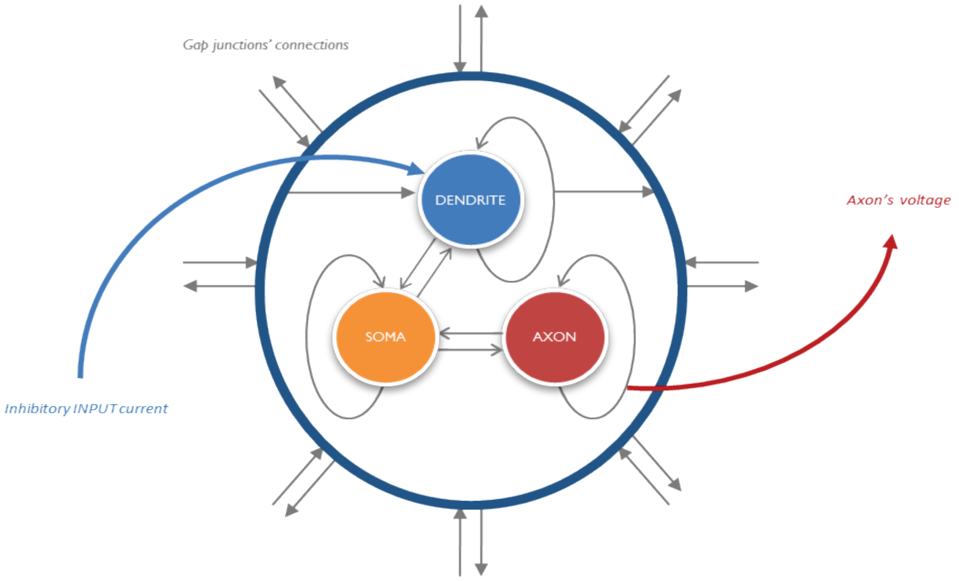


Figure 8 representation of the 3-compartmental model of a single ION cell

5.1.1 Abstract model description

The InfOli model was constructed by Jornt de Gruijl et al[12]. It is an extended Hodgkin-Huxley (eHH) neuron representation. HH models are the simplest biophysical representation of a biological neuron and one of the more prolific neuron models in the field. Simpler models that can emulate biological I/O behavior exist but represent the neuron as a black box. HH modeling is biophysically meaningful, meaning that it

[11] C.I. De Zeeuw, F.E. Hoebeek , L.W.J. Bosman, M. Schonewille, L. Witter, and S.K. Koekkoek. Spatiotemporal firing patterns in the cerebellum. NatRev Neurosci, 12(6):327–344, jun 2011.
[12] J. R. De Gruijl, B. Paolo, G. de Jeu Marcel T., and D. Z. C. I., "Climbing Fiber Burst Size and Olivary Sub-threshold Oscillations in a Network Setting," PLoS Comput Biol, vol. 8, 12 2012.

represents the actual electrochemical processes within the neuron, modeling them as RC circuits.

The ION model implements a neuron with the three basic compartments: the dendrite, the soma and the axon, as mentioned above. Within the dendrite, the model also includes the gap junctions which implement the inter-neuron connectivity within the inferior-olive nucleus. It is these GJs that complicate the model further and add the term “extended” to the standard HH model. The GJs are associated with important aspects of cell behavior as they are not just simple connections, as previously mentioned. Their significant and intricate electrical processes are reflected in their model implementation. The ION cells constantly influence each other through the GJs, leading the neurons or sub groups of neurons within the nucleus to synchronize their behavior according to the outside stimuli.

Every compartment includes a number of state parameters denoting its electrochemical state and the neuron state as a whole. The neuron state is updated at each simulation step; every new state update is based upon: (i) the previous state of the compartment updated, (ii) the previous state of the other compartments of the same neuron (mainly, the compartment voltages), (iii) the previous state of the dendritic compartment of the neurons to which the updating neuron is connected through the GJs (the dendritic voltage), and (iv) the externally evoked input to the InfOli, representing the input coming from the rest of the cerebellar circuit into the nucleus. The three compartments and GJs are evaluated/updated concurrently at each simulation step. The model is calibrated with a simulation time step of δ = 50 μs. The output to the system is considered to be the Axon voltage of each neuron per simulation step. In actual scientific experiments though every neuron state can be a potential output, depending on the subject of the experiment.

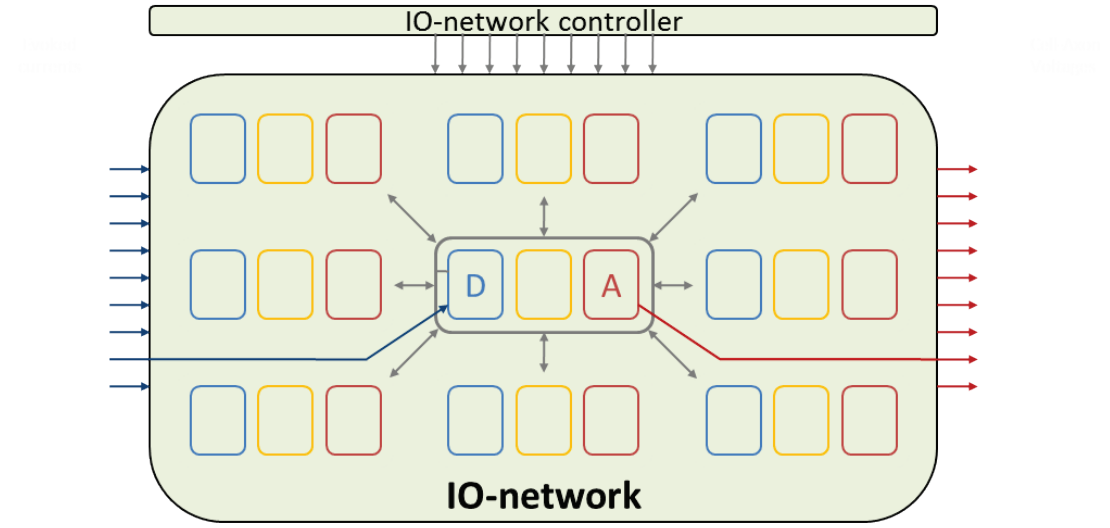


Figure 9 Representation of a 9-cell network InfOli application

Experiments practically use a number of connectivity patterns for this application. Thus, the application needs to potentially support all possible connectivities. We assume the worst-case scenario of all-to-all interconnection that also reflects that support requirement. In real deployments, the connectivity of the network would be defined using a simple connectivity matrix to be manipulated by the researchers in the field before each experiment.

A quick profiling of the C version of the application, as we can see in Table 6, reveals that the extended complexity of the neuron, and especially the presence of the GJ, makes for a quite demanding application[13].

Table 6 Neuron requirements per simulation step

Computation	**FP operations per neuron**
Gap Junction	475 per connection
Cell Compartment	859
I/O and Storage	**FP variables per neuron**
Neuron States	19
Evoked input	1 per simulation step
Neuron Conductances (cell biophysical parameters)	20
Axon Output	1 per simulation step
Compartmental Task	**% of FP ops for 96 cells**
Soma	13
Dendrite	10
Axon	8
Gap Junction	69

[13] G. Smaragdos, S. Isaza, M. V. Eijk, I. Sourdis, and C. Strydis, "FPGA-based Biophysically-Meaningful Modeling of Olivocerebellar Neurons," in 22nd ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Feb. 2014.

Already at the point of a 96-cell network, the GJs contribute 69% of the total FP operations of an all-to-all connected application. The profiling of the application reveals that, for a model that includes the GJ feature, it is the connectivity density (average percentage of the total neuron inventory to which neuron cells are connected) that dictates the computational complexity. The GJ computations increase quadratically with the network size (as the computations are repeated in every neuron, once per simulation step, for every independent connection), as opposed the linear fashion of the main neuron compartments, dictating also the overall complexity of the application.

```
for (i=0 ; i<InfOli N_INPUT; i++) {
      V = prev_Vdend - neigh_Vdend[i];
      F_new = V ? exp(?1 ? V ? V/100);
      F_acc =+ f_new;
      V_acc =+ V;
      }
Ic = CONDUCTANCE ? (0.8 ? F_acc + 0.2 ? V_acc);
return Ic;
```

Code 1 GJ computation code

Additionally, the GJ computations themselves are demanding, as they are not just passive connections, as can be seen in the GJ code segment on Code 1. These observations constitute the inter-neuron connectivity support in this application as one of the main challenges of accelerating it.

There are two distinct types of experiments that computational neuroscientists can conduct with such biophysically realistic models. One is the simulation of small-to-medium-scale networks (between 100–1000 neurons) that target real-time experimental setups (TYPE-I experiments). In this case, latency is the main constraint of the application, as the experiment would demand the step of the simulation to finish within the time span defined in the neuron model as one simulation step. The second case of experiments try to simulate large-scale networks (>1000) that begin to resemble and even reach the sizes of the actual biological systems (TYPE-II experiments). For example, the mouse inferior olive, that is one of the prime subjects of in-vivo experiments, consists of about 15,000 neurons. A cat olive has a size of 146,000 neurons while the human olive consists of around 1,000,000 neurons. In this type of experiments, population size is the main objective as well as maximizing simulation throughput. This disconnect between experimentation types is imposed mainly because of technology limitations but also due to the lack of a comprehensive and scientifically useful methods to process the amount of generated data simulation at real time.

5.2 Application-side QoS requirements

The VINEYARD neurocomputing application (a range of applications, in fact) is a traditional *batch-processing application*: A certain amount of scientific calculations representing the simulation of an arbitrarily large neural network has to be executed within a given deadline. In other words, we concern ourselves here with TYPE-II experiments since datacenter latencies (not to mention physical setups) would be unrealistic to use for TYPE-I experiments in the foreseeable future.

5.2.1 Latency

The *main unit of work* for this application are the neuron computations within each simulation step of the application, mainly the cell-compartment and the GJ computations. The simulation step of the mathematical model is set at 50 μs. This is a very stiff requirement as any other calibration leads to either a non-function model or a very unstable one. This constraint is another specific challenge, which the acceleration of the application needs to tackle. An application run that can support this constraint is considered to run the brain simulation at real-time speed. Since, for VINEYARD, only batch-mode simulations are considered, real-time latency is not required by the jobs that are to be sent to the data-center. Thus, even though this latency constraint exists within the model, there is no absolute requirement to meet it for the neuroscientific experiments to be usable, as long as the datacenter execution achieves significant speedup compared to conventional means of computing.

5.2.2 Throughput

For datacenter-based experiments, the focus is *throughput*. In this case, real-time execution is not required as most experiments are based on off-line analysis. The 50-μs latency constraint is relaxed. The problem size (network size in the InfOli case) scales up rapidly to be able to resemble or emulate the networks sizes of the biological systems (as the examples mentioned in the previous section). The FP operations required for medium-scale experiments can be seen in Figure 10.

Additionally, the conducted experiments are also long in simulated brain-time duration, often aiming at 100-200 seconds. This adds much more stress to the I/O of the application as well as to the storage requirements, as both the input and output data increase in scale. In Table 7, we can see the application requirements for experiments that would attempt to simulate the full biological counterparts of the ION.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

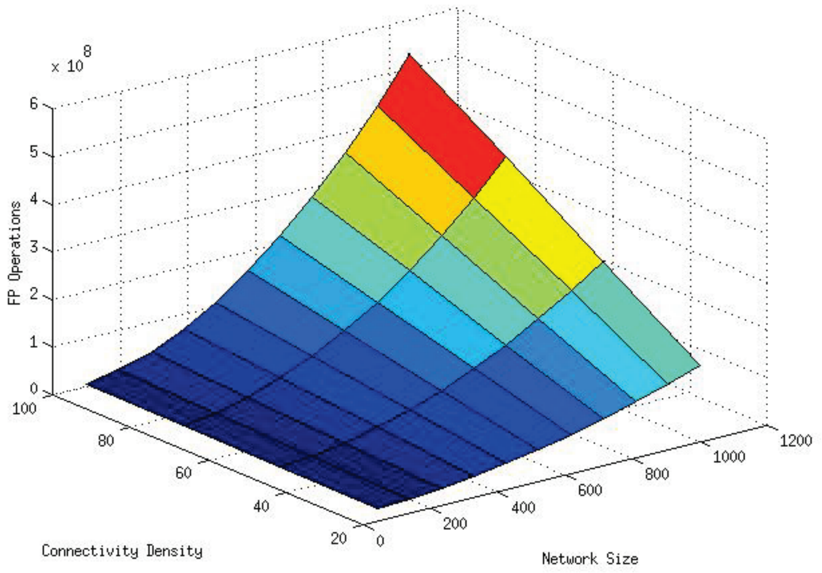


Figure 10 FP operations vs Network Size and Connectivity density for different simulation-based experiments. The worst case scenario (all-to-all) is represented by 100% connectivity density.

Table 7 Requirements for TYPE-II experiment examples emulating biological systems per simulation step

	Mouse ION	Cat ION	Human ION
Input (Gb)	0.830	79.400	3,725
Output (Gb)	0.001	0.010	0.070
Storage Registers (Gb)	0.840	79.420	3,725.4
Computation (#FP ops)	1.069E+11	1.013E+13	4.750E+14

5.2.3 Job execution time

This metric is closely related to the throughput constraints. **A job in the neurocomputing application is defined as a single experiment sent to the datacenter, simulating a certain network size and a specific amount of brain time**. Again, the requirement here is *soft*. Many of the possible experiments would require several days of simulation, using conventional means that most neuroscientists use. Reducing the required time of such an experiment to less than 24 hours would increase research efficiency greatly. Besides the obvious benefit of the simulation finishing faster, the lower simulation time makes application execution more reliable, reducing the chance of an experiment halting due to any soft or hard technical failure, subsequently reducing the need for repeated attempts.

5.2.4 Programming effort

There is no hard programming-effort constraint for this application. The limitation mostly comes from the target audience and the nature of working with such applications in computational neuroscience. Since the target audience are the neuroscientists, they will require programming frameworks that they are familiar with. Most modelers in the field work with Python-, XML-, MATLAB-based languages. A programming frameworks that is as intuitive as these or, even further integrated to these programming environments would be essential for adoption by the community. Otherwise, unless an engineering expert supports the process, acceleration setups are highly unlikely to be widely used by the neuroscientists.

Additionally, such a programing framework must be able to handle rapid changes that are constantly issued on models while executing experiments, a process that the computational neuroscientists call *model fitting*. Thus, the programming environment must not require over-optimization to provide good benefits as to keep the programming effort and, thus, the development time during fitting manageable.

5.3 Datacenter-side requirements

The InfOli application for large-scale experiments operates in batch mode. Hard deadlines for the end of experiments do not exist as all processing is to be done offline. That being said, the units/sec throughput delivery is definitely relevant as shorter experiments with higher throughput will yield the aforementioned benefits to the research process. The job in the InfOli application is defined as a single experiment run requested by the neuroscientist client, including the execution of a whole network for several simulation steps. Each job comprises tasks that execute the calculations of a subset of the network within one simulation step. **The datacenter nodes will need to synchronize after all tasks within a simulation step are completed**.

5.3.1 Power and energy consumption

Large-scale experiments would require larger simulation setups in which energy/maintenance costs become increasingly significant. A thorough analysis of the power and energy consumption and the attempt to run the InfOli application with high power and energy efficiency is constantly relevant. Such costs would significantly influence the ability of a research organization to maintain and support these experiments. That being said, there is no concrete constraint as such a limit is tightly tied to the resources and budget of the respective research organizations but defining an analytical cost model is difficult.

5.4 Motivation for application acceleration

The description and the computational demands of the application make the need for acceleration clear. It is an application that potentially has very hard latency constraints (when targeting real-time experiments) and very high throughput demands for large-scale experimentation. Indeed, the ideal platform for the model would be one that can simultaneously provides both high throughput and low latency.

Typical technology cannot cope with such demands in a satisfying way. A large part of the trend in computational neuroscience to use simpler, less accurate models than the HH or use such biophysically accurate representations on a very small scale, comes from the technical limitations of typical computing. Unless an organization has the funds to sustain a very large homogenous supercomputing cluster, truly effective experiments are difficult to conduct. Accelerator platforms can improve this point to a great extent, providing similar performance benefits to a fraction of the maintenance and energy costs, while at the same time accelerating the research process itself[14].

Moreover, the different ways that this application can be used, with so variant cases of problem sizes, experiment lengths and varied throughput demands, can benefit greatly from a heterogeneous acceleration platform. Just slight differences on these aspects can shift application behavior, thus making various accelerators viable and efficient for use, on a case-by-case basis. Thus, a heterogeneous set up that provides a variety of options is a perfect fit for this and similar classes of computational neuroscience workloads.

5.5 Tasks suitable for acceleration

The C profiling of the InfOli application reveals the GJ as the main computational bottleneck and also the aspect that dominates the computational complexity of the application. This is the first task that should be tackled through acceleration. Secondly, the main-compartment computations are also significant. Although scaling linearly with the problem size, they still include a good deal of computation. Additionally their dataflow nature gives great potential for parallelization on an accelerator platform.

Large-network experiments require also significant I/O traffic. Yet, for this specific application, no matter the problem size, the computation still remains dominant. Even though efficient data communication is useful and desirable, the main challenge for throughput are still the GJ/neuron computations as they scale up much more rapidly

[14] E. Izhikevich. Which Model to Use for Cortical Spiking Neurons? IEEE Trans on Neural Net., 15(5), 2004.

than the communication demands. Another important aspect to be considered are also the storage/memory capabilities of each accelerator, as biologically accurate experiments will certainly bring these capabilities to its limits regardless of platform.

5.6 Summary

To summarize, most of the requirements for this application are soft, as the more a datacenter can provide in terms of low latency, high throughput and energy efficiency the better. The main impact of limiting these requirements would be the size of the simulated neural networks in the experiments, leading to qualitative differences in the ability to research certain behaviors of the network explored, even though not necessarily invalidating their scientific merit.

Throughput limitations would affect experiments and the scale of the network explored, making it less biologically plausible compared to real-life counterparts. But even though a large-scale network might still be far off from biological systems, it can still be used effectively to research network behavior. Thus, a hard upper constraint on a network size that provides meaningful experiments does not exist.

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

6 Financial applications

6.1 Application description & functional requirements

This section describes the requirements of the financial use case for the VINEYARD project and its foreseen cloud infrastructure. The use case redesigns parts and critical modules of the "ATHEX Trading Machine" (ATM).

Automation using Technology in Trading & Exchanges is considered as de-facto principle nowadays. The historical "Battle of the Bund" demonstrated the value of technology in trading in a very public way[15]. But technology is getting old quickly and organizations using automation requires continuous technological change in order to stay competitive. The two prominent issues that Exchanges technology encounters these days are: **Ultra Low Latency** and **High Frequency Trading**. These two factors seem that will specify the design of the new trading environments. New technologies enable faster and more complex ways of trading, which has ultimately shifted trading activities to demand more interdisciplinary skills and ask collaborative work to be performed and deliver sufficient supporting mechanisms. This rapid, concurrent change in financial markets and technology is putting pressure on firms and Exchanges in a variety of ways.

Globalization of Exchange Market is a fact imposing standard mechanisms of communications between firms and exchanges all over the globe. By adopting FIX[16] standard, we enable the participation in the global markets by significantly reducing the cost[17]. Exchange market is hungry for a unified, global trading platform that facilitates global expansion in a risk-controlled environment. Automatic trading adds new types of risks as it misses human common sense. Protection of the market is of top priority and automated risk controls can and should feature prominently in a multilayer risk management system. Risk controls contain detection of runaway algorithms and big-finder errors as well as support of a number of automated safeguards, including volume controls, circuit breakers to halt markets in extremely volatile market conditions, and message policies to deter excessive activity by trading firms.

Concluding, our job in this project is to apply the "use the right tool for the right job" principle by choosing the appropriate technologies for demanding tasks of ATM. GPUs for the computationally intensive tasks, increasing the trading throughput and FPGAs

[15] Stephanie Hammer, Architects of Electronic Trading: Technology Leaders Who are Shaping Today's Financial Markets, Wiley Press 2013

[16] The **Financial Information eXchange** (**FIX**) protocol is an electronic communications protocol for international real-time exchange of information related to the securities transactions and markets.

[17] Source: [Public Available] FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX) Version 4.4, http://www.fixtradingcommunity.org/pg/structure/tech-specs/fix-version/44
FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX) Version 5.0 Vol 1-7, http://www.fixtradingcommunity.org/pg/structure/tech-specs/fix-version/50#Documents

for data parsing, validation and transformation as it crosses the switch – with net-zero latency penalty.

6.1.1 Overall purpose and scope

The purpose of this work is to analyze the needs for the most critical Trading Engine functions. By "Trading Engine" we mean a software mechanism that accept orders concerning sell and buy actions on instruments and match them using algorithmic trading methods producing trades according to well–defined rules. The main objective of ATM is to provide a modern and efficient engine for the trading of shares in a manner such that it will efficiently encounter high volumes of orders following extremely latency-sensitive strategies and adopting newer generation processors in order to facilitate the support of High Frequency Trading.

The final system must possess an open architecture that will enable placement of risk checks in whichever location will work best with the market's business strategy: pre-trade, at the match or post-trade. The target is to offer better control of market health by ensuring all market participants are meeting pre-defined parameters. Additionally, risk checks must cover cross-asset, cross-market analysis and be designed for low-latency so that member execution speed is not severely impacted.

The ATM eco-system includes the following major modules which communicate with each other according to the very high-level communication diagram in Figure 11.

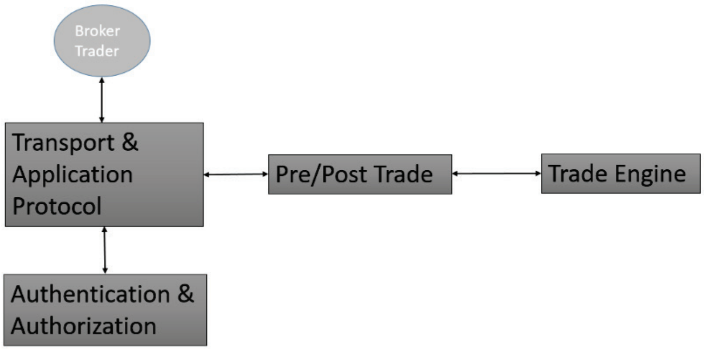

Broker
Trader
Transport & Application Protocol
Pre/Post Trade
Trade Engine
Authentication & Authorization

Figure 11 ATM Ecosystem: Main modules and Communication Links

Broker (Trader) – The users of the market – investors, banks, investment firms, etc Other users include the public and the administrators with certain roles and authorities (see Figure 12)

Transport Application Protocol- It services all routing and communication functions (gates, translations) among the external world and stock market components.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Authentication / Authorisation – Security assurance functions that guarantee validity of transactions and authenticity of users and systems.

Pre/Post Trade- Functions that prevent faults and fraud, before or after the trade. Risk management and market surveillance are among the most critical pre/post trade functions. The pre-trade risk valuation system is described in detail in Section 6.1.3.

Trading System (TS) or Trade Engine- It is the core engine that gets orders and provides trade results. Its core element is the matching engine, which is described in detail within subsection 6.1.2.

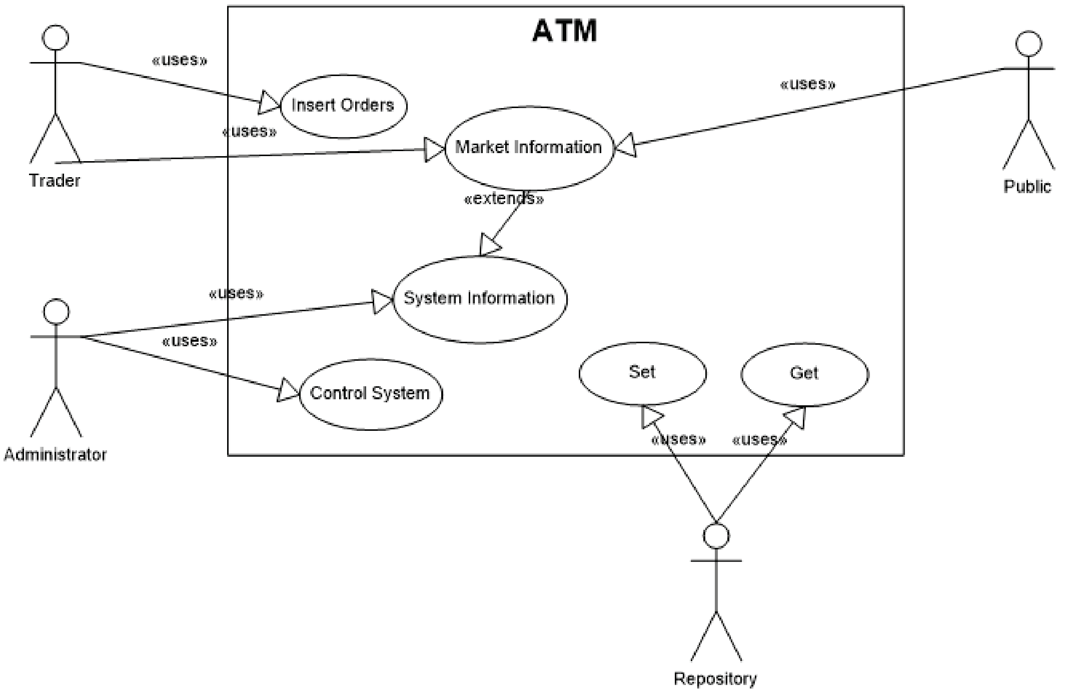


Figure 12 ATM Users and Roles

The overall ATM trading system comprises of several modules and roles, as described briefly below, in the Figure 13 below, where main processes and interfaces are depicted:

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

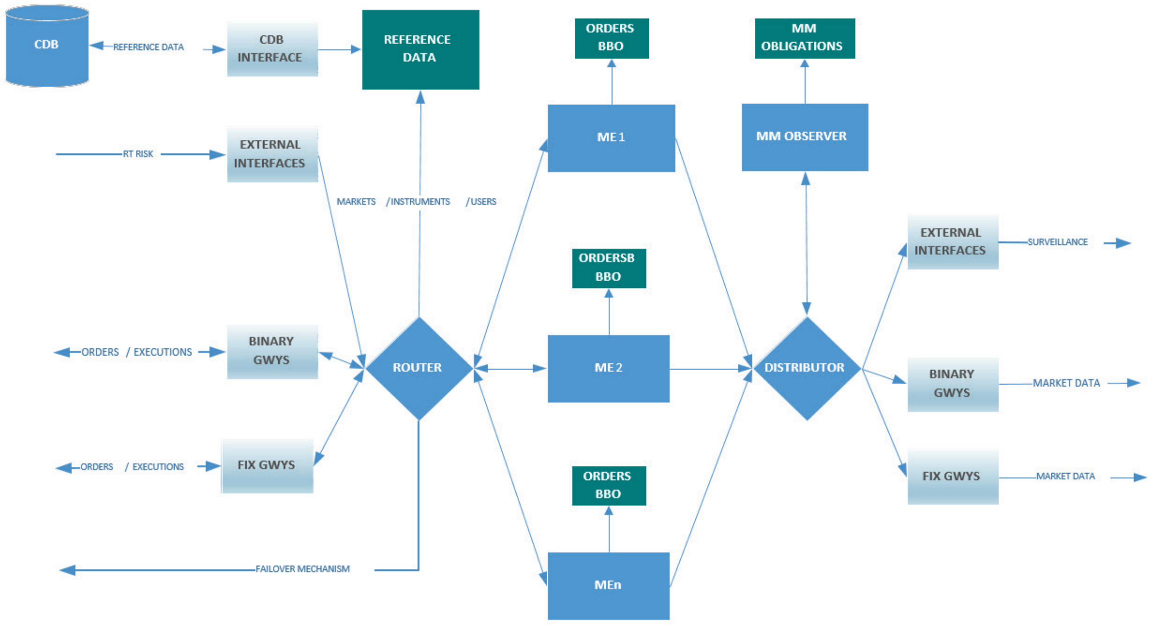


Figure 13 Trading System Overview

Processes

ME (Matching Engine) – A Number of matching engines (one or more processes each) should run in a parallel way. The number of the Matching Engines (MEs) depends on the market segmentation. The lowest level of the market segmentation is one financial instrument such as Security, Bond, and Derivative.

Router- This is the process responsible to receive all the messages and to forward them to the appropriate Matching Engine, according to the market segmentation. Also, it sends those messages to the Failover Interface.

Distributor– This is the process responsible to receive all the Market Data coming out from the MEs and distribute them to the appropriate Output Interface.

MM Observer– This is the process that is used to monitor the obligations of the Market Makers during the trading session. The input of this process is some Reference Data and the Market Data Events.

Interfaces

Binary Gateways –TCP/IP as communication protocol. Fixed-length binary format of transactions. Business rules authorization.

FIX Gateways - TCP/IP as communication protocol. FIX session layer support. Binary to FIX translation of business transactions. Business rules authorization.

CDB Interface – Oracle SQLnet as communication protocol. Used to load the reference data for the trading session (instruments, members, market rules) at system startup.

External Input Interfaces - Utilizes TCP/IP as communication protocol. Receives messages from some external systems during the trading session such as Pre-Trade Risk.

External Output Interfaces - Utilizes TCP/IP as communication protocol. Sends out information during the trading session to external systems such as Surveillance system

Market Data Binary Interfaces - Utilizes TCP/IP as communication protocol. Sends out the Market Data in a binary format.

Market Data FIX Interfaces - Utilizes TCP/IP as communication protocol. Translates and sends out the Market Data in the FIX format.

Failover Interface – Sends all the received messages to a mirror trading system in asynchronous manner. The mirror system executes the events at the same time with the primary one.

ATM's vision to grow, will encounter the need to handle a growing number of components, even reaching up to the following numbers:

- Number of Gateways: >= 3000
- Number Input Interfaces: >= 20
- Number Output Interfaces: >= 20
- Number of Instruments: >= 20.000

In such context needs, the ATM needs to be re-engineered while preserving and achieving high frequency trading. One of the functions that take part in the overall system latency are related to the Transport Application functions, namely the FIX gateways, and certainly will need to be revisited while converging to high frequency trading environments. Moreover, the most critical subcomponents, which are **the Matching Engine** and the **Pre-trade risk valuation** subsystems will be also among the ones to redesign, and will rely on accelerator components. A deeper view into them gives a better understanding for their functions, including also the interesting operations which will be targeted to improve within the project.

6.1.2 Matching engine of rule-based order-driven markets

A *market* is the place where traders gather to trade *instruments* as commons stocks, bonds, convertible bonds, warrants, etc. Trading is a search problem[18]. Buyers must find

[18] Larry Harris, Trading And Exchanges: Market Microstructure for Practitioners, 2002, Oxford University Press

sellers, and sellers must find buyers. The trading rules and the trading systems used by a market define its *market structure*. They determine who can trade; what can trade; and when, where, and how they can trade. They also determine what information traders can see about orders, quotations, and trades; when they can see it; and who can see it.

Trading takes place in *trading sessions*. There are two types of trading sessions, the continuous trading sessions and the call market sessions. In *continuous trading*, traders can attempt to arrange their trades whenever the market is open. Trading is continuous in the sense that traders can continuously attempt to arrange their trades. In *call markets*, all trades take place when the market is called. All traders trade at the same time when the market is called. Market may call all securities simultaneously or one at a time in *rotation*. Many continuous order-driven exchanges (such as ATHEX) open their trading sessions with call market auctions and then switch over to continuous trading[19]. Call is also used after trading halts (e.g. at the application of volatility interruption mechanism).

Markets produce valuable information about instrument values, transactions, who has traded, who wants to trade, and the terms on which they are willing to trade. Electronic trading systems can and produces tons of market information because all information is already in electronic form. Then, markets present some information to their traders, they sell some to data vendors, and they save most for regulatory purposes. *Market data systems* report trades and quotes to the public. Data vendors offer broadcast services and query services. *Broadcast services* provide continuous streams of information. *Price and sale feeds* and *ticker tapes* broadcast trade prices and sizes. *Quotation feeds* broadcast quotations. *Query services* produce information on demand. Users submit requests for information to the vendor’s data server.

Exchanges provide forums where traders meet to arrange trades. Traders must be members of the Exchange in order to trade. Modern Exchanges have *Order-Driven Trading Systems* that arrange trades by matching buy and sell orders accordingly to a set of rules using computers. Traders negotiate with each other only by submitting and cancelling *orders*. Orders are trade instructions that specify what traders want to trade and constitute the building blocks of trading strategies. Orders always refer the instrument (instruments) to trade, how much (*volume*) to trade, and whether to sell or buy. An order may also include conditions that a trade must satisfy. The most common conditions are on the prices that trader can accept. Other conditions may specify if trader accepts a partial fill, how long the order is valid, where to present the order, and how to search for the other side. Traders indicate that they are willing to buy or sell by making *bids* and *offers* (or *asks*). Bids and offers (*asks*) usually include information about the prices and quantities that traders will accept. The highest bid price in a market is the *best bid* and the lowest offer price is the *best offer*. A *market quotation* reports these

[19] Source: [ATHEX Corporate Document, Restricted] Η Αγορά και το Σύστημα – Γενική Διεύθυνση Λειτουργιών Χρηματιστηρίου Αξιών Αθηνών (Διεύθυνση Λειτουργίας Αγοράς & Εξυπηρέτησης Μελών), February 2013

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

best prices with the name BBO (*Best Bid and Offer*). The difference between the *best ask* (*offer*) and the *best bid* is the *bid/ask spread*.

A *market order* (*MKT*) is an instruction to trade at the best price currently available in the market. Market orders usually fill quickly, but sometimes at inferior prices because they *pay* the full *bid/ask spread*.

A *limit order* (LMT) is an instruction to trade at the best price available, but only if it no worse that the *limit price* specified by the trader. For buy orders the price must be at or below the specified limit price. For sell orders, the price must be at or above the limit price. In case the order cannot be satisfied it will *stand* as an offer and placed to a specialized structure called (*limit*) *order book*.

A *stop instruction* stops an order from executing until price reaches a *stop price* specified by the trader. Usually the stop price refers to the same instrument with the order. ATHEX also uses stop instructions on another instrument or on an index. Orders with stop instructions called *stop orders* (STOP) and can be of any type (market or limit)[20]. Stop orders became normal orders (activated) when the condition is satisfied.

There are also combination orders having more than one legs where each leg is referring to a different instrument. All legs are executed as a single transaction. Usually the number of legs is 2 and such combination orders do not exceed 5% of the total number of orders, while the trend in high speed trading is to abandon such compound orders because they are not permitting parallelism.

All prices must follow the rule of *minimum price increment* – also called the *tick* or the *minimum price variation* – and specifies the smallest amount by which two prices can differ. It is specified from Exchange and can be at instrument level.

Open orders are orders that have not yet executed or cancelled and usually reside at order book. A *good order* is an order that can be executed. All good orders are open orders.

Orders can have *validity and expiration instructions* to indicate when they are valid and when the expiry. These instructions are important for limit and stop orders i.e. for orders that do not trade immediately upon submission. *Day orders* are valid for the trading day on which traders submit them. *Good-till-cancel* (*GTC*) *orders* are valid until the trader expressly cancels them. *Good-Till-Date* (GTD) *orders* or *Good-until orders* are good until a date specified by trader. *Good-this-week* (GTW) and *good-this-month* (GTM) are special cases of Good-until orders. *Immediate-or-cancel* (IOC) *orders* are orders that are valid only when they are presented in the market. Whatever portion of the order that cannot be filled immediately is cancelled. *Fill-or-kill* (FOK) *orders* are a type of IOC orders that the whole volume must be matched or cancelled otherwise. *Market-on-open orders* (ATO) or *at-the-open* are market orders that broker can fill only at the beginning of the

[20] Source: [ATHEX Corporate Document, Restricted] ATHEX RuleBook

trading session and the *opening price* is specified by auction. *Market-on-close* (ATC) *orders* or *at-the-close* orders are market orders that broker can fill only at the close of the trading session.

Orders can have quantity instructions specifying the way the quantity filled. The most common instructions are: *All-or-none* (AON) *orders* forbids partial fill, i.e., full execution of the volume must be achieved or nor execution at all. *Minimum-or-none* or *minimum-fill* (MF) *orders* specify the minimum size of a trade.

Order-driven markets use trading rules to arrange their trades. *Rule-based order-matching systems* use trading rules to arrange trades from the orders that traders submit to them. All orders specify the maximum quantities that traders will accept. All order-driven markets use *order precedence rules* to match buyers to sellers and *trade pricing rules* to price the resulting trades.

If the market is a call market, the market collects the orders before the call. Immediately following the call, the trading system makes one attempt to arrange the trades. If the market is a continuous trading market, its trading system attempts to arrange trades whenever new orders arrive.

Rule-based order-matching system uses the same sequence of steps (algorithm) when attempting to arrange orders. They first match orders using their order precedence rules. They then determine which matches can trade. Trades will occur only if at least one buy order offers terms acceptable to at least one seller. Finally, they price the resulting trades using their trading pricing rules.

To arrange trades, order-matching systems use their *order precedence rules* to separately rank all buy and sell orders in order of increasing precedence. They match the orders with highest precedence first. The order precedence rules are hierarchical. Systems first rank orders using their *primary order precedence rules.* If two or more orders have the same primary precedence, systems then apply their *secondary precedence rules* to rank them. All rules are applied one at the time until all orders ranked.

All systems use *price priority rule* as their *primary order precedence rule*. Under this rule, priority is given to buy orders that bid the highest prices and sell orders that offer the lowest prices. Market orders always rank highest because the prices at which they may trade are not limited.

Systems (markets) use various secondary precedence rules to rank orders that have the same price. The most usually used rules rank orders based on their time of submission, on their display status and on their size. All systems must have at least one secondary precedence rule that will disambiguate the competitions. *Time precedence rule* gives precedence to the traders whose bid or offer first improves the current best bid or offer. The price-time precedence rules are the most often used rules from order-driven markets and the systems that support them called *pure price-time precedence systems.* Other types of precedence are:

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Display precedence gives displayed orders precedence over undisclosed orders at the same price. Markets give precedence to displayed orders in order to encourage traders to expose their orders. If an order is partly displayed and partly undisclosed, the market usually treats the two parts separately.

Size precedence varies from the market to market. In some markets large orders have precedence over small and in others the opposite. Most markets permit restrictions (instructions) on size of orders. Traders may specify that entire order must fulfill at once, or they may specify a minimum size for a partial execution. Orders with size restrictions usually have lower precedence than unrestricted orders because are harder to fill.

As an example, in the following table (Table 8) we present the orders submitted in an auction.

Table 8 Example of orders submitted to auction

Time	Broker	Side	Size	Price
12:01	XA	Buy	30	2.0
12:05	XB	Sell	20	2.1
12:08	XC	Buy	20	2.0
12:09	XD	Sell	10	1.98
12:10	XE	Sell	50	2.2
12:15	XF	Buy	40	Market
12:18	XG	Buy	20	2.1
12:20	XH	Sell	60	2.0
12:22	XK	Buy	70	1,98

An order book that arranges these orders by pure price-time precedence presented in Table 9. Orders with high precedence are presented at the top on the sell side and at the bottom for the buy side.

Table 9 Arrangement by price-time precedence

SELLERS			BUYERS	
Trader	Size	Price	Size	Trader
XD	10	1.98	70	XK
XH	60	2.0		
		2.0	20	XC
		2.0	30	XA
XB	20	2.1	20	XG
XE	50	2.2		
		Market	40	XF

Order matching proceeds after the market ranks its orders. In a call market that happens immediately following the market call. In continuous market it happens whenever a new order arrives. The system first matches the highest-ranking buy and sell orders to each other. If the buyer will pay at least as much the seller demands, the match will result in a trade. If one order is smaller than the other, the smaller order will fill completely. The system then will match the remainder of the larger order with the next highest-ranking order on the opposite side. If the first two orders are the same size, both will fill completely. The system then will match the next highest-ranking buy to sell orders. The process continues until the market arranges all possible trades.

In the previous example, assuming that the auction ends at 12:30, the system will arrange the following orders:

1. The system first matches XD's order to sell 10 at 1.98 with XF's order to buy 40 at the market. The system fills XD's order and leaves XF's order with a remainder of 30 to buy at the market.
2. System then matches XF's remainder of 30 with XH's order to sell 60 at 2.0. XH's order is coming next because it has the highest precedence on the sell side and now XH's order remains unfilled 30 at 2.0.
3. The system then will match XH's 30 with XG's order to buy 20 at 2.0. This match fills XG and leave XH with a remainder of 10 at 2.0.
4. System continues with XH's remainder of 10 with XA's order to buy 30 for 2.0. This match fills the remainder of XH and leaves XA with a remainder of 20 to buy for 2.0.

The next match does not result in a trade. XA's remainder of 20 to buy for 2.0 cannot trade with XB's order to sell 20 for 2.1 because XA will not pay as much as XB demands. The following table (Table 10) summarizes the trades (left) and presents the resulting order book (right) with the unfilled orders. In case that market now started continuous trading, the market quote (BBO) would be 2.0 bid for 40, 20 offered at 2.1. Continuous markets always have a spread between best bid and the best offer. If they did not, a trade would result.

Table 10 Summary of trades (left) and resulting order book (right)

SELLERS			BUYERS	
Trader	**Size**	**Price**	**Size**	**Trader**
		1.98	70	XK
		2.0	20	XC
		2.0	20	XA
XB	20	2.1		
XE	50	2.2		

Match	Seller	Buyer	Quantity
1	XD	XF	10
2	XH	XF	30
3	XH	XG	20
4	XH	XA	10
		Total	70

Single-price auctions are very common and used by most continuous order-driven stock markets, which open their trading sessions with a single price call market auction. In a single price auction, all trades take place at the same *market clearing price*. The last match that leads to a feasible trade determines the clearing price. If buy and sell orders in this match specify the same price, that price must be the market-clearing price. Matching by price priority ensures that this market-clearing price is also feasible for all previously matched orders. If the buy and sell orders in the last feasible trade specify different prices, the buy order will bid higher price than the sell order offers. The market can clear at either of these two prices or at any price between them. In the previous example, the last feasible trade is between XH and XA so the market-clearing price is 2.0.

The single auction clears at the price where supply equals demands. The orders in the limit (order) book determine the supply and demand schedules. The *supply schedule* lists the total volume that sellers offer at each price. The *demand schedule* lists the total volume that buyers want at each price. At prices below the clearing price, there is *excess demand* because buyers want to buy more than sellers offer. Likewise, at prices above the clearing price, there is *excess supply* because sellers offer more than buyers want. These schedules determine how much the market can trade at any given price. Single-price auctions maximize the volume of trade by setting the clearing price where supply equals demand. For our previous example, the supply and demand schedules presented in Table 11.

Table 11 Example of supply and demand schedules

SELLERS			**BUYERS**		
Supply Schedule	**Total Size at Price**	**Price**	**Total Size at Price**	**Demand Schedule**	**Excess Demand Schedule**
10	10	1.98	70	180	170
70	60	2.0	50	110	40
90	20	2.1	20	60	-30
140	50	2.2		40	-10
140		Any higher	40	40	-10

In order to construct the above table, first we sum the total size bid or offered at each price. In our example the only sum we have to make is that of buy side where XA and XC demand 50 at the price of 2.0. Next, sum these quantities across prices in order of decreasing price priority. Sum the supply schedule from lowest price to highest price and sum the demand schedule in opposite direction. To compute the excess demand schedule, subtract the supply schedule from the demand schedule at every price. The following figure (Figure 14) displays the two schedules and the crossed value at the market-clearing price of 2.0.

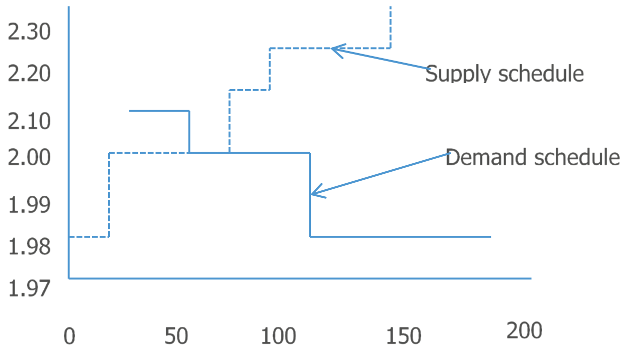


Figure 14 Demand and Supply schedule: The crossed value

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

6.1.3 Pre-trade system – Risk valuation

When an option trade is made between a buyer and a seller, the seller has made an *obligation* to buy or sell the underlying at the strike price at some time in the future. The buyer has the *right* to sell or buy the underlying at the strike price at some time in the future. In the case of a forward trade, both the buyer and the seller make an obligation to buy or to sell the underlying at some pre-determined price at some time in the future. One of the principal functions of a **clearing house** is to guarantee that all contracts traded will be honored[21]. This means that the clearing house becomes the counterparty in all transactions. This means that buyers and sellers of derivatives acquire rights and obligations with respect to the clearing house, not to the original counterparty (Figure 15).

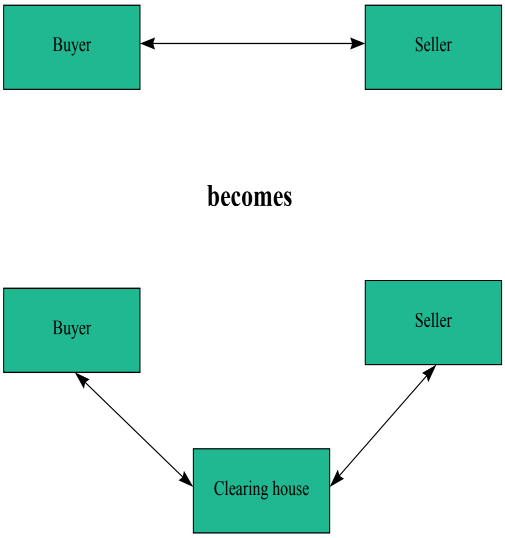


Figure 15 Rights and Obligations

This guarantee function eliminates any questions about the creditworthiness of the original counterparty, or the ability to fulfil his or her commitments. The clearing house guarantees all transactions. All market participants must have complete confidence in the ability of the clearing house to guarantee fulfilment of all obligations. Confidence is based on the existence of three main aspects: membership criteria, the margin system and the capital resources of the clearing house, including risk insurance.

[21] Source: [ATHEX Corporate Document, Restricted] Κανονισμός Εκκαθάρισης Συναλλαγών Επί Κινητών Αξιών σε Λογιστική Μορφή, Ιανουάριος 2015

The clearing house is taking the risk that a participant - a member or a client - may fail to fulfil his obligations after an unfavorable change in the price of the underlying security. If this happens, the clearing house is obliged to neutralize the position of the failing participant. This means that the clearing house must repurchase written[22] positions and resell held[23] positions. This may be at prices that far exceed those in effect when the participant took his positions.

In order to guard the clearing house against losses in such cases, each participant having an obligation has to pledge collateral. This collateral is pledged to a bank - either directly or indirectly via a member. The collateral can be cash, stocks, bonds or other financial instruments that are acceptable as collateral. In most cases, only a certain percentage of the market value is accepted as collateral. This collateral may be taken over by the clearing house if the participant fails to fulfil his obligations[24].

The amount of collateral that has to be pledged is called "*margin requirement"* or just "*margin*". The margin requirement should be the cost of immediately neutralizing an account. This should in theory be the negative market value of the account. However, an account cannot normally be closed at the instant the participant defaults and at the prevailing market prices. It can take time to neutralize the account, and the value of the account can change during this period. The purpose of a margin requirement system is to calculate the "true" margin requirement for each account, which should equal the maximal possible cost for neutralizing the account, according to a number of assumptions. The margin requirements should not be too small, since the clearing house may then lose money. Neither should they be too big, since this may discourage trading. The procedure of calculating the margin requirements is called **margin calculation**[25]. The sub system used for margin calculations within the Clearing System is called **RIVA (Risk Valuation) or Pre-Trade Risk System (PTRS)**.

Figure 16 below, illustrates the Risk Valuation system,

(1) taking initially **Reference Data** from the clearing system where dematerialized securities reside and from the trading system. Reference Data include
   - Markets Rules and Instruments from the trading system (named OASIS).
   - Clearing Accounts, Positions & Collaterals from the clearing system (Dematerialized Securities named SAT)

(2) receiving continuous **Dynamic Data** from the trading system. Dynamic Data include Orders, Trades & Market Data from the trading system. During the trading day as the **system** will constantly receive the market events from the matching

[22] Written position: The number sold of a specific product (also called short)
[23] Held Position: The number bought of a specific product (also called short)
[24] Source: [ATHEX Corporate Document, Restricted] Αναδιάρθρωση Υπηρεσιών Εκκαθάρισης, Διακανονισμού και Καταχώρησης στην Αγορά Αξιών ΧΑ - Σχεδιαζόμενες Αλλαγές στα Πληροφοριακά Συστήματα ΟΑΣΗΣ και ΣΑΤ - Έκδοση 1.4 Αθήνα, Μάρτιος 2010
[25] Source: [ATHEX Corporate Document, Restricted] The Margining Subsystem Methodology Guide

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba
European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

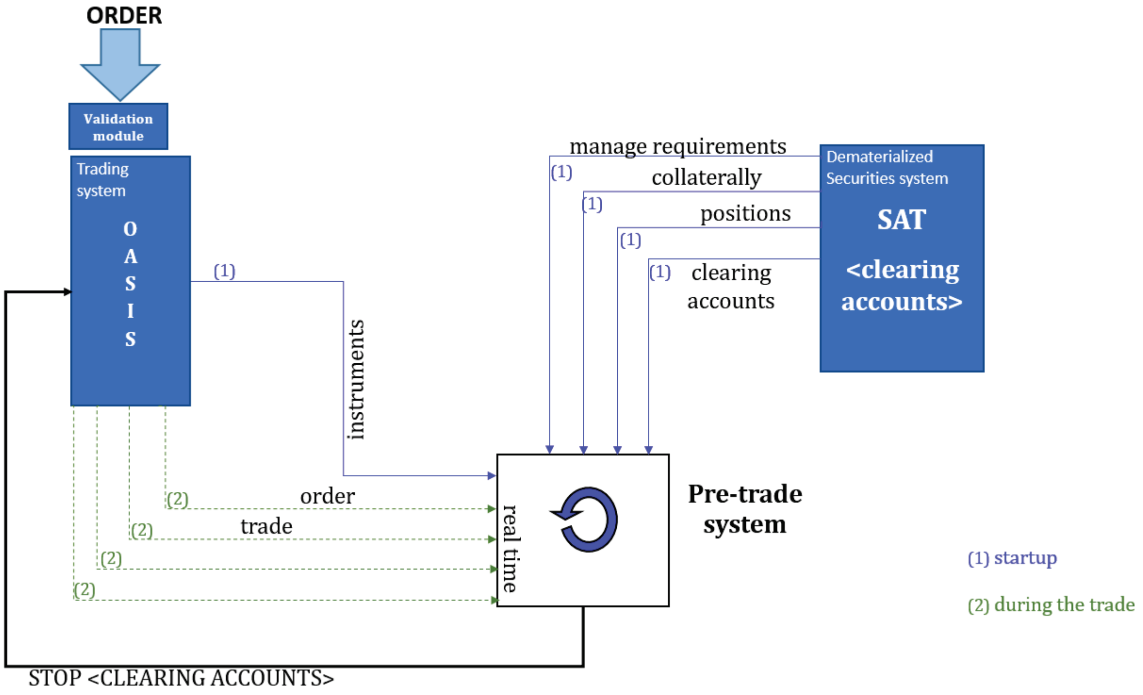
engine such as the execution reports (orders and trades) and the market data (last sale information, BBO, etc.).

(3) performing **Risk Valuation**, by evaluating the positions (production of vector files) and the collaterals per clearing account. The Pre-Trade Risk Valuation system will evaluate the investor's portfolios (net long or short position) and will compare the results against the credit limits set by the CCP and clearing firms. Binomial, Black & Sholes evaluation models for the derivatives market are invoked.

(4) producing **Blocking Message** when and if required –If a calculated value exceeds the specified limit, the risk valuation system will send a message to the matching engine in order to block the ordering for a specific clearing account. The purpose of such limits is to enable clearing firms to prevent customers from accumulating positions that exceed levels at which the clearing firm is financially comfortable.

(5) More over and before the entry to the matching engine, the **validation module** will examine the quantity or the value of individual orders and it will either accept or reject it. The opposite happens either when the margin is increased or the position risk is reduced.

(6) The almost real-time evaluation of the positions on derivative products triggered by events like last sale of the underlying products will cause the recalculation of the positions of every clearing account related to those specific products. Also the mathematic formulas used for option's evaluation like binomial and Black & Soles require big processing power of the system. Low latency and high performance should be the main characteristics of the risk valuation system.



PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba
European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

Figure 16 Pre-trade risk valuation

The basic algorithmic functions which are employed in this continuous process are the following:

- Black And Scholes (standard) and/or specifically Black-76 model
- Binomial trees
- Vector file calculation

Risk Valuation formulas - When calculating margin requirements for options, Risk Valuation (RIVA or Margining Subsystem) uses theoretical formulas for pricing options in each valuation point. A number of different formulas exist; risk valuation uses the Black&Scholes, Black -76 and binomial methods. In those methods, the price of an option depends on the following:

- Underlying price – the price of the underlying in this valuation point.
- Strike price – the strike price of the option.
- Risk-free interest rate – is considered constant and is an input parameter.
- Volatility level.
- Time to expire for the option.
- Dividends – known or expected dividends for the underlying affects the value of the option.

The following table (Table 12) shows which method Risk Valuation uses for different types of options:

Table 12 Formulas for Option valuation

Product	**Method**
American call based on spot prices	Standard Black&Scholes (year 1973)
American put based on spot	Binomial trees without dividends
European opt based on future	Black -76

Vector file production – Before explaining the contents and the role of a vector file, certain definition are made below for the following: lead time, valuation interval, valuation points and volatility.

Lead Time - The margin requirements are normally calculated once a day. It can take time to neutralize a position, therefore it may not be possible to neutralize an account at the moment the participant fails to provide the required collateral. As a result, there is a lead time from the moment collateral has been provided until the clearing organization is able to close the participant's account. The length of the lead time depends on how long it takes to discover that the participant has not provided enough collateral, and how long it takes to neutralize the account.

Valuation interval - To determine the maximum neutralization cost for the account, possible values of the account must be calculated for the duration of the lead time. Since the value of the account is determined mainly by the price of the underlying security, it is important to know how much this price may fluctuate during the lead time. Calculating the possible prices of the underlying security results in a valuation interval. The size of the valuation interval depends on the length of the lead time and the size of the historic fluctuations in the price of the underlying security over such a period. Technically, the size of the valuation interval is set as a risk parameter in the reference database. It is normally given as a percentage of the last paid price for the underlying.

Example:
Assume that the valuation interval for an index is ±10 % of the closing index. Based on a closing index level of 1200, the calculation would be as follows:

```
Upper limit:  1200 + (10 % x 1200 ) = 1320
Lower limit:  1200 - (10 % x 1200 ) = 1080
```

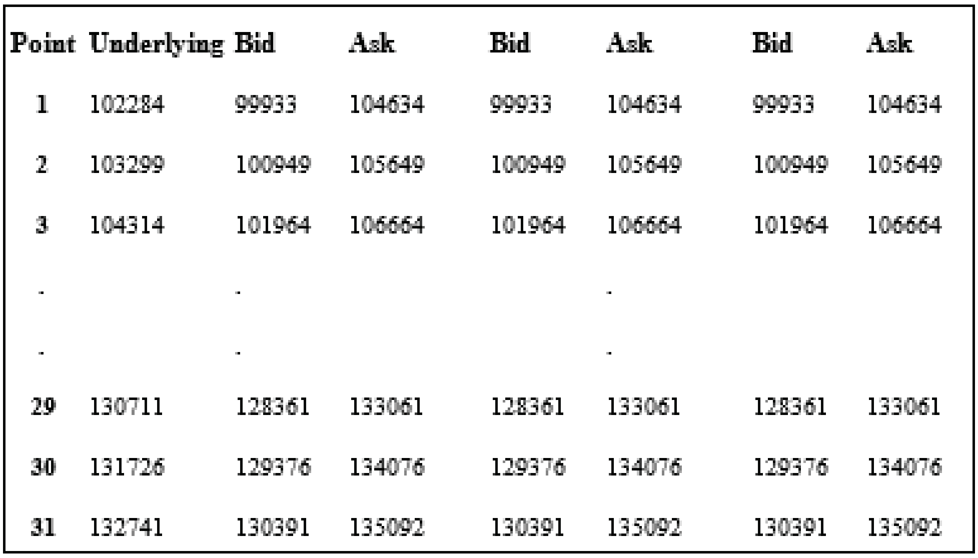
Therefore, risk valuation would calculate margins based on the premise that the index will not go below 1080, or above 1320 during the time period (the lead time).

Valuation Points - The upper and lower limits of the valuation interval represent the extreme movements allowed for the calculation. However, the risk profile of options is such that the “worst case” cost of neutralizing a portfolio containing different options and futures based on the same underlying value, can occur anywhere in the valuation interval. Therefore, the valuation interval is divided into a number of valuation points, where 31 is a typical value. The closing price of the market represents the mid-point, and in case of 31 valuation points, it has 15 valuation points on either side of it. At each valuation point, the cost of neutralizing the position is calculated, based on the value of the underlying in this particular valuation point. The margin requirement will be the worst case of the neutralizing cost of all the valuation points.

Volatility - The price of an option can also be strongly affected by changes in volatility. The risk of changed volatility is taken into account by calculating the value of the account, based not only on the current volatility, but also on a higher and a lower volatility. The amount with which the volatility is increased or decreased is determined by risk parameters. **Thus, the neutralizing cost is calculated at each of the valuation points for three different volatility levels. A typical valuation interval therefore consists of 3 X 31 valuation points.**

Vector file – The important result entity of risk valuation is the "vector file". It is a file consisting of series data that is shared by all positions in the series. The exact definition is product dependent. For each contract that risk valuation handles, as many calculations as possible are performed on an instrument basis, without using details from individual positions. This data is calculated per instrument, valuation point and held or written position. The main reasons for vector files are: 1) Computational efficiency and 2) The need to distribute vector files externally, in order to enable members to replicate the system and to calculate their own margins as well as those of their customers.

By adding position data to the vector file, we get the neutralizing cost in each valuation point for a single position. The term "positional vector file" is used for this. Figure 17 below could represent a vector file.



Point	Underlying	Bid	Ask	Bid	Ask	Bid	Ask
1	102284	99933	104634	99933	104634	99933	104634
2	103299	100949	105649	100949	105649	100949	105649
3	104314	101964	106664	101964	106664	101964	106664
.		.			.		
.		.			.		
29	130711	128361	133061	128361	133061	128361	133061
30	131726	129376	134076	129376	134076	129376	134076
31	132741	130391	135092	130391	135092	130391	135092

Figure 17 A Vector-file example

This is a matrix and when we talk about points we mean one cell in the matrix. In one dimension the underlying price is altered and in the other dimension the volatility. If this instrument is not effected by volatility, i.e. Forwards or Futures, the values for the different volatilities will be the same. There are two values in each column that correspond to held and written positions (bid/ask differences). The components in the vector file are multiplied by 100 because they are internally treated as integers, i.e. in practice we have two decimals in the calculations.

Calculation of cells in the vector file, involves formulas of variant types depending on the type of contracts: forwards, futures, options, stock lendings and whether it is a single or cross-margining procedure. An indicative example for vector file calculations is found below. The example is about the case of futures, where the cell calculation in the vector file is a linear calculation, while for options Black-76 and/or binomial algorithms are invoked.

Example: Calculating a Vector file for futures[26]

Variables	*SP* - Last paid price for the underlying (spot price). *P* - Number of contracts in the portfolio
Instrument Data	*CM* - Contract size. The number of instruments that defines one contract for an instrument
Risk Parameters	V_u Upward valuation interval. This represents, in percent, the maximum increase in price for this underlying during the lead time. V_d Downward valuation interval. This represents, in percent, the maximum decrease in price for this underlying during the lead time. AF_B Adjustment factor held. In order to reproduce the bid/ask spread, an adjustment term is calculated by multiplying the spot price by this factor. For calculations on bought contracts, the point values are reduced by this adjustment term. AF_S Adjustment factor written. In order to reproduce the bid/ask spread, an adjustment term is calculated by multiplying the spot price by this factor. For calculations on sold contracts, the point values are reduced by this adjustment term.
Margins	RM_B Required margin for bought contracts. RM_S Required margin for sold contracts.

$$RM_B = \left(\pm \frac{V_{u/d} \cdot SP}{100} - \frac{AF_B}{100} \cdot SP \right) \cdot CM \cdot P$$

$$RM_S = \left(\pm \frac{V_{u/d} \cdot SP}{100} + \frac{AF_S}{100} \cdot SP \right) \cdot CM \cdot P$$

The result is rounded to the nearest integer. The vector file is a simple linear function. The valuation interval is divided in the same way as for forwards:

[26] Source: [ATHEX Corporate Document, Restricted] The Margining Subsystem Methodology Guide

Upper interval:

$$subinterv. = \frac{SP}{50(n-1)}V_u$$

Lower interval:

$$subinterv. = \frac{SP}{50(n-1)}V_d$$

$$\boxed{\begin{array}{lcc}
1 & -\frac{V_d \cdot SP}{100} - \frac{AF_B \cdot SP}{100} & -\frac{V_d \cdot SP}{100} + \frac{AF_S \cdot SP}{100} \\
2 & -\frac{V_d \cdot SP}{100} + 1\frac{SPV_d}{50(n-1)} - \frac{AF_B \cdot SP}{100} & -\frac{V_d \cdot SP}{100} + 1\frac{SPV_d}{50(n-1)} + \frac{AF_S \cdot SP}{100} \\
 & \vdots & \vdots \\
n-1 & +\frac{V_u \cdot SP}{100} - 1\frac{SPV_u}{50(n-1)} - \frac{AF_B \cdot SP}{100} & +\frac{V_u \cdot SP}{100} - 1\frac{SPV_u}{50(n-1)} + \frac{AF_S \cdot SP}{100} \\
n & +\frac{V_u \cdot SP}{100} - \frac{AF_B \cdot SP}{100} & +\frac{V_u \cdot SP}{100} + \frac{AF_S \cdot SP}{100}
\end{array}}$$

We get a $n \times 2$ matrix where bought futures are represented by the first column and sold futures by the second. The two columns are duplicated, in order to have values for the different volatility levels that are represented in the vector file.

$$\boxed{\begin{array}{lcccccc}
1 & Bid & Ask & Bid & Ask & Bid & Ask \\
2 & Bid & Ask & Bid & Ask & Bid & Ask \\
3 & Bid & Ask & Bid & Ask & Bid & Ask \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
n-1 & Bid & Ask & Bid & Ask & Bid & Ask \\
n & Bid & Ask & Bid & Ask & Bid & Ask
\end{array}}$$

This matrix is the vector file for futures, and is used as a starting point for margin calculations for all positions on this future.

With this vector file, there is no need to bother with data for the instrument. The following calculation is performed in all points for the individual position.

$$RM_{B,i}(VF(i)) = \frac{VF(i)}{100} P \cdot CM$$

$$RM_{S,i}(VF(i)) = -\frac{VF(i)}{100} \cdot P \cdot CM$$

$RM_{X,i}(VF(i))$ is the margin requirement if using the vector file value in point $VF(i)$.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

By using these formulas, it is possible get a positional vector file for this position. This is used as input for Cross Margining[27].

6.2 Application-side QoS requirements

The financial applications are online-processing applications, both latency and rate sensitive: Zero-latency requirement is of utmost importance for the matching engine functions, while high-throughputs are also required as demand is growing and is not stable along the day session. Message translation latency is also critical to reduce as it affects user responsiveness as well as the ability to accommodate high frequency trading.

6.2.1 Main functional requirements

A list of ATM's functional requirements is written up below, either in the form of general requirements or specific to the trading system and the pre-trade risk valuation system.

General Functional Requirements

- Support of multilingual data fields, to be encoded using the UNICODE character set (UTF8).
- The system must take into account the current hardware and network infrastructure. The system must not affect at once the whole infrastructure. This means that at the first phase of production, the system must be capable of replacing the existing system without further changes at the firm's site. Protocol conversions at transport and application levels must be applied in order to encounter these issues.

Trading-System Functional Requirements

- Support multiple markets with different procedures and trading rules.
- Ability to have different schedules (hours of trading) per market.
- Support of multiple currencies, i.e. allow securities trading in different currencies.
- Ability to compute stock exchange indices by taking into account securities from different markets.

[27] Cross Margining: The vast majority of accounts contain positions of 2 or more types of contracts (e.g. 2 different strike options), where the risk of the position is the combined risk characteristics of the different contracts registered to the account. Cross margining is when the margin calculation takes into account the off-setting characteristics of the instruments

- Support of Attribution and Anonymity Order Entry, i.e. the firm can specify if the orders sent by its members are treated as anonymous or attributed as specified by the firm.
- Support of multilevel transaction audit mechanism. Storage formats and access Application Programming Interfaces (APIs) must facilitate data visualization and transmit compliance.
- Support of different matching algorithms, i.e. techniques to allocate matched quantities.
- Support of four types of interfaces: Order Entry, Market Data, Post-Matching/Clearing, Control.
- Support of Financial Information eXchange (FIX) Protocol (http://www.fixtradingcommunity.org/) semantics, i.e. message names & types, field names & types, data types, and relationships between them.

Pre-Trade Functional Requirements

- Support of FIX standard as the primary interchange mechanism with remote clients and Order Routing Systems (ORS).
- Wide variety of risk checks available for margin checks, daily quantity checks and “fat finger” checks.
  - Prevent the entry of orders that exceed appropriate pre-set credit or capital thresholds in the aggregate for each customer and the broker-dealer.
  - Prevent the entry of erroneous orders, by rejecting orders that exceed appropriate price or size parameters, on an order-by-order basis or over a short period of time, or duplicative orders.
  - Apply a set of Risk & Volatility Mitigation Controls as: Protection Points for Market & Stop Orders, Maximum Order Size Protection, Cancel on Disconnect Protection, etc.
- Centralized Drop Copy mechanism to send copies of execution reports, heartbeats and acknowledgements, and trade bust messages through a FIX protocol-based messaging interface[28].

6.2.2 Execution time

The financial application is an online processing type application, where transaction rate and transaction latency matters. Total execution time for an order stream is not of interest, though the execution time of each transaction unit which participates within the overall latency of a transaction is critical.

[28] Source: [Public available] FIX Adapted for STreaming(SM) FAST Protocol(SM), http://www.fixtradingcommunity.org/pg/structure/tech-specs/fast-protocol

In our cases, current indicative execution times include:

Matching Engine Execution time	~10ms
B&S execution time:	0,061 msec
Black 76 execution time:	0,063 msec
Binomial execution time:	1,226 msec

The goal is to achieve significant reduction of the execution time for the abovementioned steps plus the FIX translation and parsing time, at a level which enables the whole trade cycle for a single order including the risk assessment steps to conclude within a real-time context that is defined at the order below 1 μsec. However, final requirements will be defined after a more thorough study on the profile of the involved algorithms and processes.

6.2.3 Latency

Latency requirement of a new ATM is one of the most important, hard, constraints. Technology options must be leverage proven. Minimizing latency while maximizing protection is the most challenging requirement related to matching engine and risk valuation efficiency. Credit & Risk Controls must be implemented in hardware-accelerated solutions at a fraction of the associated latency of software-only solutions, thus somewhat eliminating any potential concern about speed versus managing risk that we currently have. It is desirable to be able to support "near real time" availability without manual intervention and transparent to the foreign applications.

Latency should be less than 1 millisecond

Of course, the latency can be measured within various points of the whole order flow process. Speaking about sub-millisecond requirements, latency is measured at the matching engine entry point up to the final response of the system about a matching transaction, whether done (trade) or not. However, the requirement for sub-millisecond order of delay remains, even at the level of the trader when its system is collocated at trading system's premises, and the transport application circuit functions (FIX gateway and order routing) are performed in the same HPC environment with the matching engine (i.e. an HPC cluster with InfiniBand connectivity).

Today's single-processor infrastructure, and the overheads imposed by the processing and transport of long messages, achieve an overall latency of 10 milliseconds, while the matching latency is 3 milliseconds.

6.2.4 Throughput

ATM's throughput requirements are summarized below, at various levels of work definition (order, session of one trader, trade day):

- **Number of Trades/Orders per day >= 100 million**
- **Session throughput: >= 1000 orders/sec**
- **System throughput: >= 100.000 orders/sec**

Today, matching engine operations run in a single processor environment, which limits scalability, exhibits an average system throughput of 300 orders/sec. Therefore, the demanding requirements cannot be satisfied unless parallelization and acceleration is achieved.

6.2.5 Power & energy consumption

The project aims to adopt state of the art hardware technologies in order to gain a technology edge, featuring high performance gains and scalability with low consumptions patterns. The new ATM modules will be assessed in terms of power and energy consumption. An analysis for the consumption patterns will be appropriate so that the effect to actual budget performance can be accurately estimated. Efficiency will be measured based on metrics such as:

- Transactions / Watt
- Transaction / Joule
- It is expected that power consumptions will increase in the new accelerated environment; however the factor of transactions per energy unit should be highly improved, exhibiting cost-efficiency at high performance levels.

6.2.6 Programming effort

The new system must possess an open architecture that will enable easy placement of modules when that is necessary. It is necessary to keep change management being efficient to perform, as operational rules and new add-ons might be necessary; critical components of the systems such as the Matching Engine and the Pre-Trade risk valuation should be designed to behave as service boxes, providing their services transparently to the rest components of the ATM. Familiar open software communication structures will be utilized so that not specialized resources will be necessary for other forthcoming adaptation that might take place.

Software architecture should be modular, flexible and scalable adapting modern massive parallelism technologies using hardware accelerators for heavy duty tasks. Specifically, software processing will rely on modern cost effective hardware customization technologies of FPGAs, huge and cheap "off-the-shelf" processing power of GPUs and integrate them with new multicore CPUs. Only by adopting state of the art software and hardware technologies, ATM can gain a technology edge, featuring high performance gains and scalability with low consumptions patterns.

Obviously, the project requires adopting a modern software development environment suitable to reach the maximum hardware efficiency suitable for both client & server applications. The environment must cover two basic requirements. A) A modern Integrated Development Environment supporting the full Software Life Cycle and B) Offer a development sound methodology for the software administration. Among the required characteristics are: Version Control, Professional Profile & Debugging applications, OS Independence, etc.

Upon this environment, flexibility and cost efficiency should be assessed in terms of #(man-effort hours) multiplied by #(hourly rates) for each personnel category. Achieving high levels of abstractions at the programmer's level, will not ask for high expertise and specialized skills into developing efficiently, high performance and scalable algorithms and processing modules in parallel environments, as accelerating farms resources will take over this responsibility and it will only require the programmer's lines to invoke these resources in a proper way for the necessary computations. Not any current indicator is existing to quantify this requirement. However, the resulting efficiency will be measured in various experiments to show the estimated resources in terms of time and money.

6.2.7 Cost efficiency

A significant goal is related to the financial benefits that is achieved after the deployment of the new ATM based on modern accelerating components. Financial analysis should quantify:

- The risk costs today and the cost benefits by reducing the level of risk exposure
- The investment costs for the purchasing new technologies that will be required and for extra developments
- The difference into operating costs which are due to changes into maintenance costs, programming efforts for operation/support/maintenance, energy consumption patterns.

Different ICT infrastructure versions should be assessed, but the criterion should incorporate not only the volume of processing that is being achieved within a certain operation expenditure limit, but also all financial parameters, including financial gains, investment costs and human resources costs, which will be necessary and affected from the new technology.

6.3 Datacenter-side requirements

Data center requirements are not hard. Application instances will first run onto Neurocom's machines with realistic datasets before migrating them onto partners' infrastructures for the final assessment. ATM will be supported by a modular, flexible and scalable architecture adapting modern massive parallelism technologies using hardware accelerators for heavy duty tasks. Specifically, it needs to adopt modern cost effective hardware customization technologies of FPGAs, huge and cheap "off-the-shelf" processing power of GPUs and to integrate them with new multicore CPUs.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

A view of data-center resource requirements for the current operations can be seen in the following table.

Table 13 An example of data-center resources required for application execution

Resource (for project duration)	Total	ATM (one matching and two communication servers)
Computing resource (Core Hours per year)	560.000	1720 (35 for matching)
Persistent storage (TB)	186 (2x93)	0,922 (0,59 for matching)
Data to be backed up (TB/month)	270	5,12 (1,02 for matching)
Amount of Data to be transferred to computing Centre or (TB/month)		0,092 (0,08 for matching)
Memory (GB)	2.240	76 (12 for matching)
Power consumption (KW)	137,6	0,51 (0,23 for matching)

6.4 Motivation for application acceleration

The rapid, concurrent change in financial markets and technology is putting pressure on firms and Exchanges in a variety of ways. But rather than view technological change as a problem, agile organizations are turning to technology as a source of competitive advantage; the field programmable gate arrays (FPGAs), graphical processing units (GPUs), microwaves, and cloud computing highlighted are delivering edges in speed and flexibility. For example, the application of GPUs brings order-of-magnitude performance increases for compute-intensive trading and risk management applications, and similar cost savings over conventional computing grids. This performance increase allows for better assessment of trade ideas and superior quantitative research in pre-trade area. On the other hand, an FPGA can be programmed to perform many tasks in parallel offering the ability to have many, many instructions that can be carried out in a single clock cycle, which accounts for the speed advantage. FPGAs excel at simple data transformation operating directly on a physical electronic signal as it comes through a cable on a server. This simply means that server can consume data as fast is delivered (link speed).

The main arguments supporting the acceleration of various components within ATM application are related to business facts and operational objectives.

a. The core trading functions need to feature the smallest latency possible, so that high speed trading can be supported. This also adds to stock market competitiveness and sustainability. Trading at high frequencies needs sub-millisecond latency at the matching engine, in order to server the order streams as well as high speed updating in order to reliably update high speed trader systems. Scaling becomes also important, in order for ATHEX to consider its position in the future, towards addressing the needs of multiple markets in its geographical area. In that context, re-engineering the trading systems focuses towards supporting radical increase of order volumes and rates, as well as, increase of numbers of external interfaces, actors, traders and market viewers.

b. And as the trade cycle time is shortened because of improved architectures and powerful implementations (including the accelerated matching engine procedures), the more difficult it gets for Pre-trade risk calculations to complete their results in the case of derivatives. Pre-trade risk calculation needs to be accelerated enough, so that to be able to in-time block asynchronously risky transactions. Now, and for the derivatives, such ability is not achieved, and the need to speedup risk calculation and to pro—actively take measures becomes obvious. From the business point of view, the stock exchange market organization, keeping the role with the clearing obligation, needs to settle efficiently any risks sourced by derivatives (financial or commodities). This is not possible under current infrastructure and conditions, and becomes even harder to achieve within an accelerated trading environment.

6.5 Tasks suitable for acceleration

The tasks of the application that can be accelerated are highlighted below:

a. *FIX message parsing/validate/transform/format*: FIX protocol is text based variable length application protocol designed mainly for financial applications. Due to its textual nature, it requires parsing before the data will be used and formatting before data send to the network. Parsing and formatting are inherently sequential procedures and can slow down the whole process.

b. The FAST (Fix Adapted for STreaming) protocol was designed as a way to reduce the bandwidth and network-latency required to distribute market data without incurring excessive CPU costs. The protocol defines various fields and operators which are used to identify specific stocks and their pricing. An important aspect of FAST is its compression mechanism which reduces bandwidth, however, introduces significant CPU overhead. In fact, decoding of FAST represents a major bottleneck which makes it particular interesting for offloading to an FPGA.

c. *Computation of Auction Price:* Auction of a board can be scheduled to occur during trading session but can also happen under extreme market conditions (volatility out of range). Auction can be applied to groups of instruments or atomically to one instrument (volatility interruption mechanism). It is important to be able to compute auction prices for all instruments at the same time in order not to lag when changing from auction to continuous trading.

d. *Indices.* The computation of market indices is real time process and can negatively affect the matching engine by stealing processing time.

e. *Security counters (Volume, # trades, High, Low, …):* Every trade produced by the matching engine creates a number of collateral computations on a number of counters concerning information about the securities participated in the trade.

f. *Market indicators:* Market indicators visualize aspects of the market. Similarly with indices are real time computations based on the trades. These computations must not steal processing time and slow down the speed of matching engine.
g. *Continuous Automatic Matching Model (CAMM):* In case we do not have STOP on different symbol orders (GPU/FPGA). Continuous trading is the most demanding processing phase in exchange. Its speed specifies the efficiency of matching engine and if it can be fulfil the needs of HFT (High Frequency Trading)
h. *Computation of Close Price:* Computation of Close Price can be considered as a case of auction price computation.
i. *Credit Control at Pre-Trade module:* Main benefits include the ability to block orders for derivatives prior to their execution, based on the capability to complete the execution of risk calculation, having achieved acceleration of Black & Scholes, binomial and/or vector files calculation (RIVA methodology).

6.6 Summary

The current environment for the financial services in ATHEX is considered poor with certain limitations. Therefore, the most that the new components can provide in terms of latency, throughput and energy efficiency is a benefit. Limitations in latency will affect the real time achievable risk checks at the pre-trade phase. Throughput limitations would affect the market sizes which can be efficiently services. Both factors are of extreme importance towards ATHEX becoming a highly competent financial-services provider in the area. It is helpful enough the fact that normal orders are all enabling parallelized processing within the trading system operations.

The level of achievement in respect to latency might be limited by the limits of parallelization, as imposed by certain combinatory orders – now less than 10% of the total orders. However, by reducing the level of participation (or by even eliminating their presence at all) into the traffic of orders, one can maximize the benefits of the foreseen implementation.

Table 14 Financial applications summary table

Requirement	Low	Average	High
Latency (us)	10000	1000	500
Throughput (orders/second)	500	50.000	100.000
Power consumption (W)	-	-	-

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

High throughputs will exploit parallelism which will increase power demands in comparison to existing power consumptions. However, increase of power needs will be accompanied with a remarkable growth of performance and abilities. Another important benefit is related to the fact that CPU-intensive roles and very-low latency transactions processing will be assigned to appropriate components which will take care for the performance; hence, as demand will grow, programmer's efforts and concerns will concentrate onto their best integration rather than on continuous development and re-engineering of their applications.

Considering more futuristic market conditions, especially in highly volatile markets such as energy markets and after wider merging of activities and international cooperation, requirements will become even more demanding, as indicated by the Table below:

Table 10 Financial applications summary table indicating future requirements

Requirement	Metric
Latency (us)	1
Throughput (orders/second)	10.000.000
Power consumption (W)	Same as today

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

7 Transactional-analytics applications

7.1 Application description & functional requirements

Databases are a central component at any current information system. This means that the component most frequently found in a data center is precisely the database. It turns out also to be one of the most computationally intensive applications as well. This means that its efficiency is key to increase the efficiency of datacenters.

In the database world, traditionally, two workloads have been identified: On Line Transactional Processing (OLTP) and On Line Analytical Processing (OLAP). OLTP lies in the workload received by operational databases that include a high fraction of updates and consists of mainly of short queries and updates. The OLAP workload is quite different from OLTP. It consists of large queries traversing millions to billions of rows. OLAP systems do not have updates, although they have a load phase to introduce the data from operational systems to be analyzed.

Databases so far has been centralized systems with an inter-woven code covering the different functionalities, query processing, storage management, and transactional processing. However, this monolithic design resulted in systems that could not scale out and was difficult to make databases to scale up. In fact, only IBM DB2 managed to scale up to large levels in the IBM zeta mainframes.

With the advent of NewSQL, a wave of new solutions have been proposed to scale out transactional databases. Out of all them, the only solution that can scale today to very large levels (100s to 1000s of nodes) is LeanXcale. The main reason is that LeanXcale has conceived an innovation that enables to scale-out transactional processing in a linear fashion. With a scale-out transactional processing, scaling out query processing and storage management is relatively accessible.

Databases have been possibly the only software systems where benchmarking has been used systematically to compare the different systems available in the market. The Transaction Processing Council (TPC) was founded to create, evolve and certify such benchmarking processes. There are different benchmarks for different kinds of systems. The two most relevant are TPC-C[29] for OLTP systems and TPC-H[30] for OLAP systems.

More recently, database systems are emerging with combined capabilities enabling them to satisfy a hybrid workload for OLTP and OLAP. LeanXcale is one of such database systems. There is currently no benchmark available at TPC for hybrid workloads, but there is an initiatives to provide such a benchmark, called CH-Benchmark[31]. It combines TPC-C and TPC-H taking advantage that they share the use case and most of the tables.

[29] Source: [Online Available:] http://www.tpc.org/tpcc

[30] Source: [Online Available:] http://www.tpc.org/tpch/

[31] Benchmarking & Simulation Florian Funke, Alfons Kemper, Thomas Neumann. Benchmarking Hybrid OLTP&OLAP Database Systems. pp. 320-409. BTW. 2011.

7.1.1 TPC-C

TPC-C as aforementioned is a benchmark for OLTP workloads. The benchmark has been modeled after a wholesale parts supplier that operates out of a number of warehouses and their associated sales districts. The benchmark scaling is performed keeping some invariants. Each warehouse supplies ten districts, and each district serves three thousand customers.

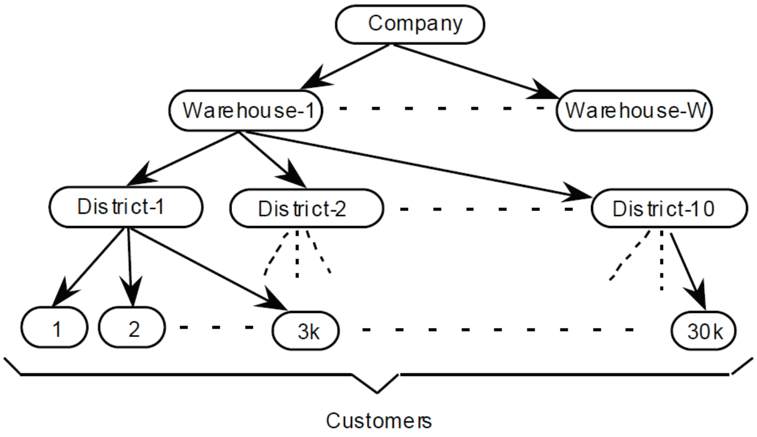


Figure 18 TPC-C Scaling

The database schema of TPC-C is depicted in Figure 19. The figure also depicts the relative cardinalities of the tables for the scaling factor. W stands for the number of the warehouses that is how the load is scaled.

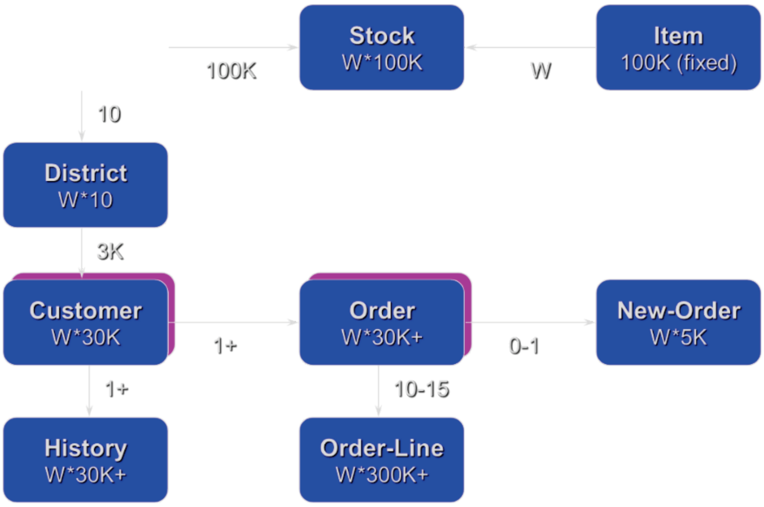


Figure 19 TPC-C Schema

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

The application executes five kinds of transactions:

- New-order: enter a new order from a customer.
- Payment: update customer balance to reflect a payment.
- Delivery: deliver orders (done as a batch transaction).
- Order-status: retrieve status of customer's most recent order.
- Stock-level: monitor warehouse inventory.

The two most frequent transactions are new-order and payment. New orders contain 10 items on average. 10% of the new orders require getting items from more than a warehouse. This avoids trivial sharding solutions in which a sharded database instance simple serves the transactions related to a subset of warehouses. The actual transaction mix is shown in Figure 20. The Payment transaction records the payment from a customer. Both new-order and payment transactions update a row for the data warehouse. This is a source of write-write conflicts if transactions run for too long (when they run for longer than 0.5 seconds conflicts become very frequent). The delivery transaction is a prototypical transaction for doing batch processing, it processes a batch of 10 orders for delivery. The order-status transaction checks the status of the most recent order performed by a particular customer. Finally, stock-level is a relatively heavy read-only transaction that queries the system for potential supply shortages by checking the level of stock at the local warehouse.

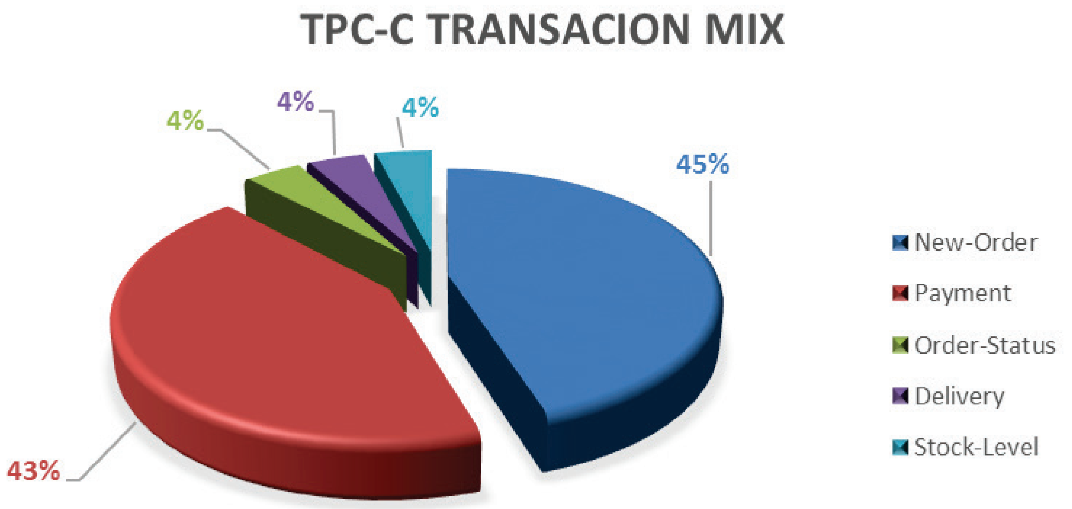


Figure 20 TPC-C Transaction Mix

The concurrency level is scaled also based on the number of warehouses. Basically, there are ten terminals per warehouse submitting transactions concurrently. They have a cycle in which they have a keying time, they submit the query and then they have a thinking time before cycling again.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

The specified times are:

	Key Time	Thinking Time
New Order	18	12
Payment	3	12
Order-status	2	10
Delivery	2	5
Stock-level	2	5

7.1.2 TPC-H

TPC-H is a benchmark for analytical workloads. The benchmark has not been designed to model a particular industry or particular business segment. Instead, it models a business that sells/distributes a product worldwide. The benchmark distills the stereotypical analytical queries performed stressing the database query engine in different dimensions common across all industrial sectors. The queries answer business questions around these issues. There are in total 22 queries that provide answers to the following classes of business analysis:

- Pricing and promotions.
- Supply and demand management.
- Profit and revenue management.
- Customer satisfaction.
- Market share.
- Shipping management.

The database schema of the database is illustrated in Figure 21.

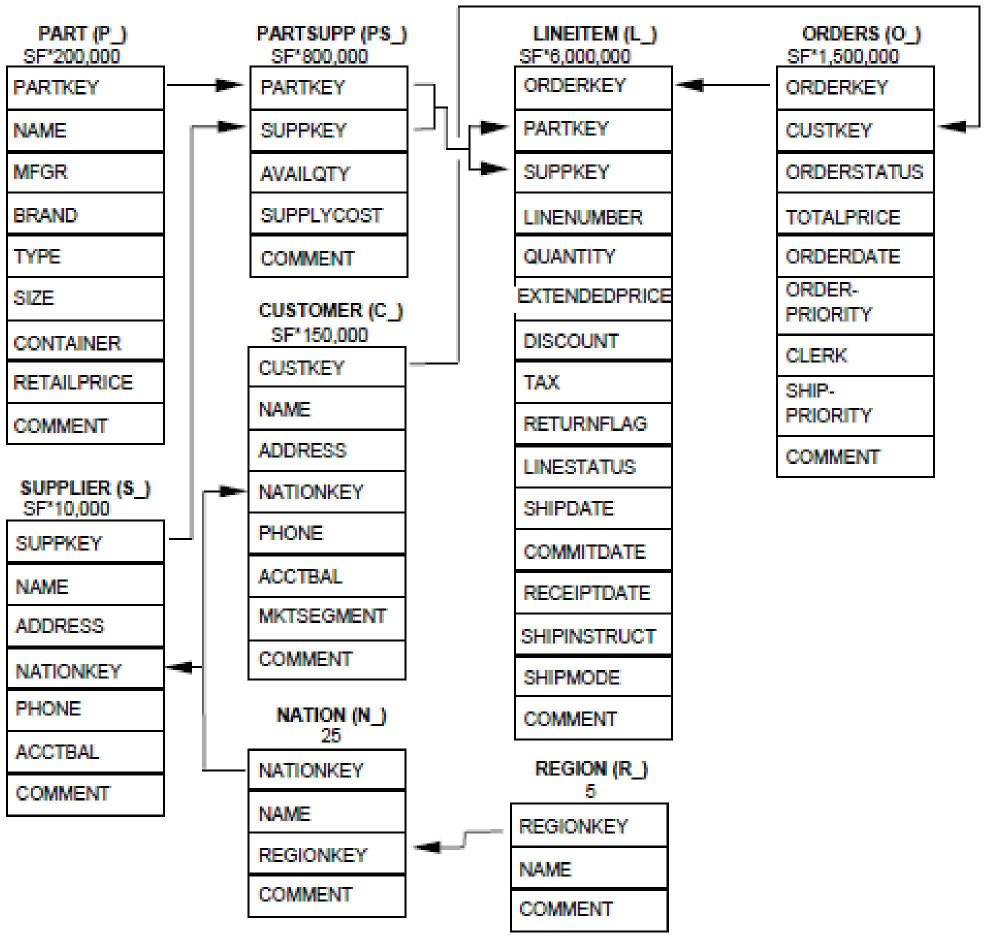


Figure 21 TPC-H Database Schema

The scaling of TPC-H is depicted in Figure 22. It is based on a "Scaling Factor" (SF) that can be chosen out of these values: 1, 10, 30, 100, 300, 1000, 3000, 10,000, 30,000, 100,000. Roughly each unit in this factor corresponds to 1 GB of data, therefore going from 1 GB to 100 TB.

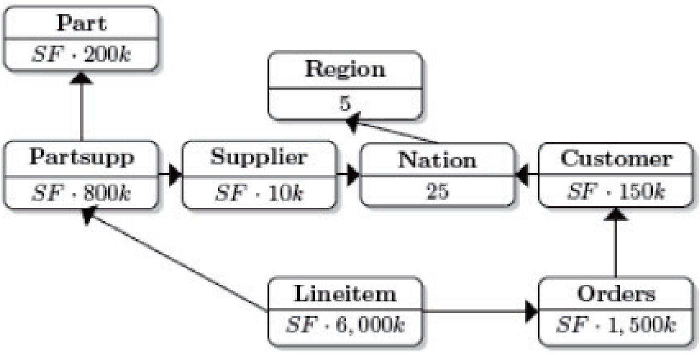


Figure 22 Scaling of TPC-H

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

The benchmark measures the response time of the 22 different queries for a given number of concurrent clients that cycle over the whole set of queries. The clients do not have any waiting time between query submissions. The benchmark also measures the loading time of the data.

7.1.3 CH-Benchmark

TPC-C and TPC-H focus on two extremes of a spectrum. TPC-C focuses on OLTP high update workloads. TPC-H focuses on OLAP workloads with heavy queries. However, there is a lack for a benchmark that can evaluate systems with both capabilities, that is, the capability of answering OLAP queries over operational data. The proposal of CH-Benchmark aims at covering this gap. Since LeanXcale aims at provide this functionality it will also be an important benchmark to measure the improvements brought by the acceleration by VINEYARD into LeanXcale and the business applications running on top of it.

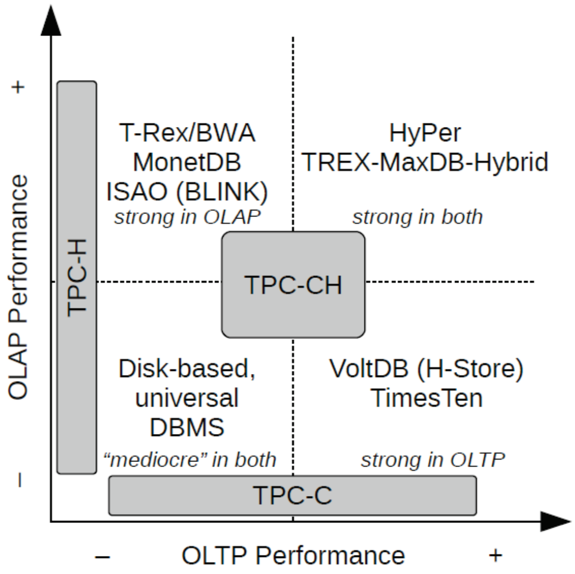


Figure 23 Spectrum of TPC benchmarks and positioning of CH-Benchmark

Since TPC-C and TPC-H have significant similarities and both do an excellent job at evaluating their target workloads, CH-Benchmark proposes to combine both of them. The identified similarities are that they model businesses related to product distribution, and therefore deal with orders and customers, as well as the details of orders such as items and order lines. The differences lie in some tables that exist in one of the benchmark, but not in the other and vice-versa.

The CH-Benchmark introduces all tables from both benchmarks and combines them in a single schema. The schema and the scaling factor is shown in Figure 24. The scaling factor has been unified by relaying on the TPC-C scaling factor, the number of warehouses.

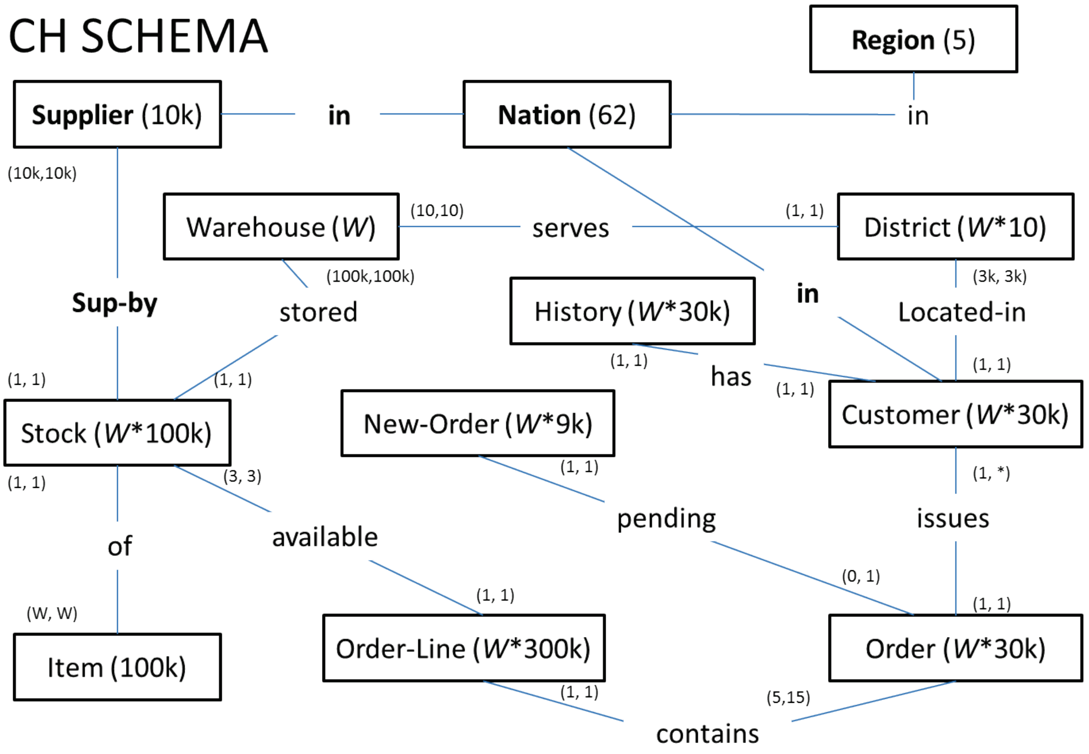


Figure 24 CH-Benchmark database schema

The injected load is the combination of injecting the target TPC-C load plus a number of TPC-H clients that cycle over the adapted TPC-H queries.

The benchmark still needs to be improved in some aspects. LeanXcale team is working in that direction and has already proposed some improvements in [Pereira].

7.1.4 Micro-benchmarking

In order to be able to measure more exhaustively the benefits of the acceleration brought by VINEYARD into some functions of LeanXcale, some micro-benchmarks will be performed to evaluate some of the accelerated functions.

One of the expected functions to be accelerated is conflict management. For this, a micro-benchmark will be created. The micro-benchmark will only evaluate the subset of components involved in conflict management, namely the conflict managers and the local transactional manager.

Other functions that we aim to accelerate with VINEYARD are compression and encryption, very important functions for data management that are CPU intensive and for which acceleration can bring significant improvements. The compression is expected to be applied at the storage engine level to store data pages in a compressed manner and reduce the IO bandwidth. Different kinds of compression techniques will be considered among Lempel-Ziv-like algorithms, Tries, dictionary encoding and Huffman compression.

7.2 Application-side QoS requirements

7.2.1 Execution time

Since all benchmarks are related to online applications, the execution time is not relevant. There is one exception that is the loading of the database in the case of TPC-H and CH-Benchmark. This is a job that is executed and for which the main metric of interest is the execution time, that is, the time it takes from the start of the loading processing till its completion.

7.2.2 Latency

One of the most relevant metrics of database systems is the latency of queries. The latency measures the time it takes since the application submits the query till it receives the answer to the query. The different benchmarks exercise very different workloads and therefore, have very different values on the latency from milliseconds to minutes. In the case of TPC-C there are SLAs established over the latency. The limit in latency for the different transactions is summarized in Table 15.

Table 15 SLA for latencies in TPC-C

	Latency (seconds)
New Order	5
Payment	5
Order-status	5
Delivery	5
Stock-level	20

This SLA is imposed over the 90-th percentile of the different transactions. For TPC-H, the latency of queries is measured in the power test, in which a single client executes one by one the queries.

7.2.3 Throughput

The throughput is another of the major metrics for database management systems. It can measure transactions per unit of time (in TPC-C, and CH-Benchmark) or queries per unit of time (in TPC-H, CH-Benchmark and YSCB). In the case of TPC-H and YCSB there is no SLA over the throughput. However, in TPC-C the throughput should lie between 9 tpmC (new order transactions per minute) and 12.86 tpmC per warehouse.

In TPC-H the throughput is evaluated in the throughput test in which a number of concurrent clients execute depending on the scaling factor as reported in Table 16.

Table 16 TPC-H Throughput test

SF	#Concurrent Clients
1	2
10	3
30	4
100	5
300	6
1000	7
3000	8
10000	9
30000	10
100000	11

The throughput metric of TPC-H is TPC-H Composite Query-per-Hour Metric (QphH).

European Commission Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

7.3 Datacenter-side requirements

7.3.1 Power efficiency

For the power consumption, both TPC-C and TPC-H provide a standardized method to measure it. The specification for the power consumption benchmarking is captured in TPC-Energy. For TPC-C the unit of power efficiency is Watts/KtpmC, that is, Watts per throughput. For TPC-H the unit of power efficiency is Watts/KQphH, power consumed per throughput.

TPC-Energy aims at extending TPC benchmarks with a power efficiency metric. In most cases, the benchmarks are online systems and therefore, the metrics for measuring the outcome of the benchmarks are throughput for a given configuration. In particular, for LeanXcale, since it is an online database, all relevant benchmarks.

7.3.2 Cost efficiency

Both TPC-C and TPC-H provide a metric for cost efficiency. In both cases, the metric computes the cost over three years of hardware, software license and support. For TPC-C the cost efficiency metric is *TPC-C Price/Performance ($/tpmC)*. For the TPC-H the cost metric is *TPC-H Price/Performance ($/QphH)*.

7.4 Motivation for application acceleration

Data management is one of the pervasive functions in any data center. Most business applications rely on an operational and/or analytical database to perform their function. This means that a high fraction of the power consumed at a data center is consumed by the database. Optimizing the power efficiency of databases will result in a very significant reduction of energy consumption not able to be achieved in other application domains.

7.5 Tasks suitable for acceleration

Within the LeanXcale database there are several functional blocks depicted in Figure 25. The major functions are:

a. **Transactional manager.** It is in charge of providing data consistency guarantees in the advent of failures and concurrent access to the database.
b. **Storage engine.** It is in charge of storing data in persistent storage, provide caching and basic functions for managing data such as insert, update, delete, search the value associated to a key, scan a range of primary keys, and even apply a filter to do a selection during a scan of range keys and/or perform a project of the columns to be recovered.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

c. **Query engine.** It is in charge of interpreting queries and executing them by interacting with the storage engine. Transactional semantics is enforced by interacting with the transactional manager.

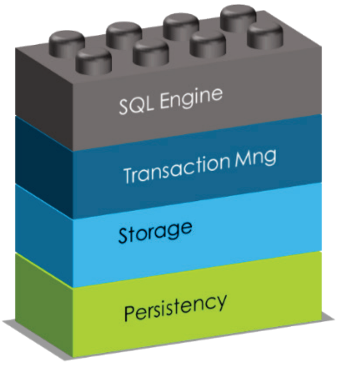


Figure 25 LeanXcale Functional Blocks

Out of these three functions we have examined which are the ones more suitable for acceleration. We discuss briefly the analysis performed for each of the functions. The transactional engine performs several tasks summarized in Figure 26.

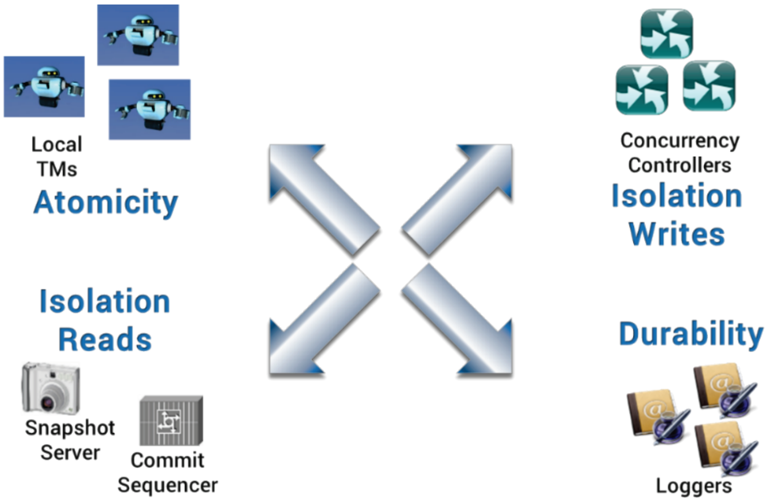


Figure 26 LeanXcale Transactional Functions

- **Conflict management**. Checks and detects write-write conflicts between concurrent transactions.
- **Logging**. Stores in persistent storage the updates to guarantee durability of transactions.

PARTNERS
Institute of Communications and Computer Systems, Maxeler Technologies, Bull Systems, Queen's University of Belfast, Foundation for Research and Technology, The Hartree Centre / Science and Technologies Facilities Council, Neurasmus BV, Neurocom Luxembourg, Hellenic Exchanges SA, Holding, Clearing, Settlement and Registry, LeanXcale, Loba

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

- **Commit sequencer**. It is in charge of providing commit timestamps for committing new transactions.
- **Snapshot server**. It is in charge of determining what the current snapshot is by observing what transactions are readable and durable.
- **Local Transactional Management**. It takes care of the lifecycle of transactions interacting with all other transactional components.

The task that seems more suitable for acceleration is the conflict management. It requires processing messages with many conflicts and then perform the conflict check. Conflict managers basically receive three kinds of messages. The first one is related to conflict checking. Each message contains a batch of hashed keys. They are checked in a hash table for potential conflicts with updates over the same key by a concurrent transaction. Since all data is of fixed length and a repetitive task that can be performed in parallel, it is quite suitable for FPGA acceleration.

Another task that can be potentially accelerated is the snapshot server. It requires processing messages and combine contiguous intervals. It keeps internally, the current snapshot that is the commit timestamp for which all previous commit timestamps have been reported (either as durable & readable or discarded), and thus, there are not gaps, and a set of commit timestamp intervals. The snapshot server receives from local transaction managers intervals of commit timestamps. It has to find which interval is consecutive with the current snapshot to advance it. In order to reduce memory utilization and advance work it can also merge contiguous intervals as a single larger interval. This task also uses data of uniform types and lengths and can be performed in parallel and therefore seems suitable for FPGA acceleration.

The storage engine is in charge of the following tasks:

- Perform Inserts, Gets, Updates and Deletes over the data based on the primary key.
- Perform range queries applying projections and/or selections.
- Store data persistently, possibly compressed and/or encrypted.

The storage engine offers two really good opportunities for acceleration. One is the compression and another is encryption. A very important fact about LeanXcale storage engine, KiVi, is that the writing can be done asynchronously to the persistent media with an arbitrary delay. This means that pages can get frozen to be written to disk and in background being compressed and/or encrypted by the acceleration framework without consuming any host CPU and at a high rate due to the inherent parallel processing of the accelerators.

Compression is not yet implemented in LeanXcale due to the CPU overhead. Now with FPGA acceleration it seems that it will become affordable since it will not consume host CPU. For this purpose different kinds of compression will be implemented and accelerated by means of the VINEYARD acceleration framework. In particular, the following kinds of compression will be tried: Lempel-Ziv family of algorithms, Trie trees, Hoffman encoding, etc.

The third function is query processing. Our initial thoughts were that analytical queries can benefit from acceleration since they involve large volumes of data. Unfortunately, some queries are ad-hoc, there are from time to time new queries deployed in an exploitation system what requires to solve queries in a short period of time what is kind of incompatible with FPGAs that take very long to get flashed. Also the processing with GPUs has a relatively expensive process to be prepared. For this reason, query processing has been, at least in a first stage, not considered for acceleration. We will consider at a later stage some specific kinds of applications such as geographical applications that require geographical queries that might exploit GPUs and image and video processing that might also exploit GPU capacity.

Having said this, there is a function related to query processing that is the computation of statistics that is required for optimizing queries. These statistics mainly compute the histogram for each index. The statistics can also be computed in the background. This task might also be suitable for acceleration and will also be considered for FPGA acceleration.

7.6 Summary

Database management systems and the business applications running on top of them are online applications with strict SLAs. There are some regular functionalities of a database management system that can be accelerated and will be focus within VINEYARD to accelerate such functions within LeanXcale such as conflict management. There are some other functionalities nice to have such as compression and encryption that are quite CPU intensive and being able to perform them through an accelerator without consuming host CPU becomes an important incentive to use accelerators.

On the other hand, the new storage engine of LeanXcale database has a design that is very appropriate to perform many functions without being in the critical path of the response time to the client, what enables to perform these functions through the accelerators. This will be another of the primary goals within vineyard.

We are considering new requirements not considered initially in the DoW to add new functionality to LeanXcale to support encryption and compression that are two critical functionalities not provided today by LeanXcale, that are very CPU eager, and that can be accelerated with the VINEYARD framework.

8 Cloud computing applications

Besides the three real-world applications, we are also going to demonstrate the advantages of the VINEYARD framework in several widely-used cloud applications based on widely-used programming frameworks such as Spark. In this chapter, we present the type of cloud applications that will be studied and the main requirements of these applications.

8.1 Application description & functional requirements

Cloud-computing applications can be divided into two broad categories as was described in Section 2:

- **Batch-processing applications (Offline):** In this case, the applications process high volumes of data that have been collected and stored in the data centers. Usually there are several complicated processing that needs to be done in the data. The main performance metric in these applications is the throughput and the execution time. In this category fall several applications like:
    - **Data Analytics:** In this case, massive amounts of data (Big Data) needs to be processed in order to extract useful information. Typical applications are text classification, book recommendations, and spyware detection. A typical framework for the development of these application is the Apache Mahoot framework.
    - **In-Memory Data Analytics:** In these cases, the data are preferably stored in the memory through Resilient Distributed Datasets (RDD). Typical applications in this category are the applications that are based on the Apache Spark.
    - **Graph Analytics:** In this case belong applications that are mainly working on parallel distributed graph processing. Typical Graph applications are graphs of social networks and web graphs. The most widely used framework for these applications is the Apache GraphX framework.
- **Streaming processing applications (Online or Real-time):** In this case, the applications process high volume of streaming data and usually the processing that needs to be done in these cases is simpler than in the case of batch processing applications. The main performance metric in these applications is the latency (i.e. N-th percentile latency). In this category falls several applications like:
    - **Data Caching**: A typical framework for these applications is the memcached framework that is used to cache data in memory.
    - **Data Serving**: These applications are generally used of online services that rely on NoSQL data stores. A typical framework is the Cassandra framework.

European Commission | Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

- **Media Streaming**: Media-streaming applications must be able to sustain both high throughput and low-latency details. A typical application that is used to run media streaming is the Nginx server.
- **Web Search**: The typical performance metric for web search is throughput (searches/sec) and (low) latency.
- **Web Serving**: In this category belong all web-service applications usually deployed by a web server, a cache server and a database server.

In VINEYARD, we plan to study the acceleration of the Spark framework since it is one of the most widely-used framework for data analytics. Spark has been adopted widely in recent years for big data analysis by providing a fault-tolerant, scalable and easy to use in-memory abstraction. Specifically, Spark provides programmers with an application programming interface centered on a data structure called the resilient distributed dataset (RDD). RDD is a read-only multiset of data items distributed over a cluster of machines, which is maintained in a fault-tolerant way. It was developed in response to limitations in the MapReduce cluster computing framework, which forces a particular linear dataflow structure on distributed programs. MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs function as a working set for distributed programs that offers restricted form of distributed shared memory. Therefore, the latency of such applications, compared to Apache Hadoop, may be reduced by several orders of magnitude.

When the user runs an *action* (like collect), a **Graph** is created and submitted to a directed-acyclic-graph (DAG) scheduler. The DAG scheduler divides the operator graph into (map and reduce) stages.

A stage is comprised of tasks based on partitions of the input data. The DAG scheduler pipelines operators together to optimize the graph. The final result of a DAG scheduler is a set of stages. The stages are passed on to the Task Scheduler. The task scheduler launches tasks via a cluster manager. The Worker then executes the tasks for the task processing as is depicted in Figure 27, below.

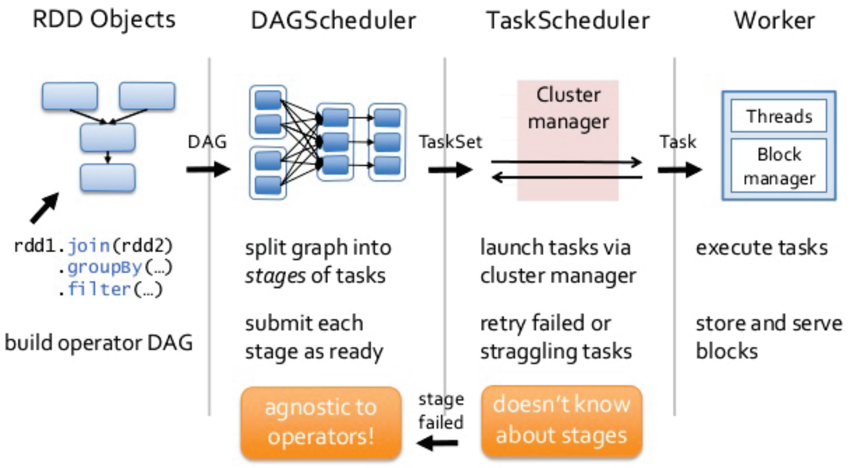


Figure 27 Spark framework

Spark libraries cover 4 main categories of applications: machine learning, graph computation, SQL query and streaming applications.

- Spark MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives.
- GraphX is a Spark API (Application Programming Interface) for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing the Resilient Distributed Property Graph: a directed multi-graph with properties attached to each vertex and edge. GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.
- Spark SQL provides the capability to expose the Spark datasets over JDBC API and allow running the SQL like queries on Spark data using traditional business intelligence (BI) and visualization tools.
- Spark Streaming can be used for processing the real-time streaming data. This is based on micro batch style of computing and processing. It uses the DStream which is basically a series of RDDs, to process the real-time data.

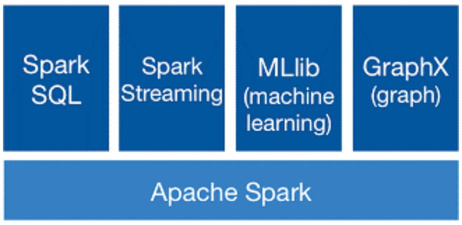


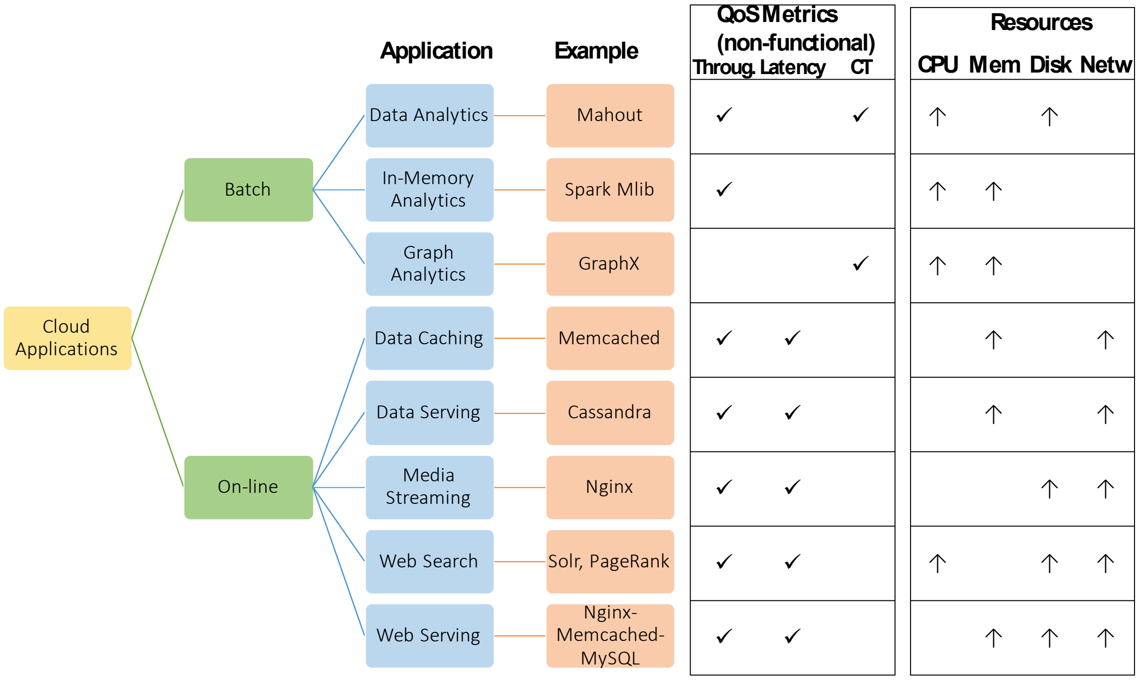
Figure 28 The Spark libraries

Several application using the Spark framework will be analyzed in order to find the hotspot and the main bottlenecks. The main functions that are computational intensive will be mapped to hardware acceleration units in order to speed up the total execution time and to improve the throughput and the latency.

8.2 Application-side QoS requirements

Figure 29 depicts a classification of the cloud applications and typical frameworks that are used for each application. For some applications, like data analytics and in-memory analytics, the main metrics for the evaluation of the performance are the processing throughput and completion time (CT). However, for other applications like data caching and data serving, besides throughput, another main metric for performance evaluation is latency. The figure shows also if the application is mainly CPU-intensive, memory intensive, Disk I/O intensive or Network I/O intensive. Each of these application may have specialized libraries for specific cloud applications.

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628



	Application	Example	QoS Metrics (non-functional)			Resources			
			Throug.	Latency	CT	CPU	Mem	Disk	Netw
Batch	Data Analytics	Mahout	✓		✓	↑		↑	
	In-Memory Analytics	Spark Mlib	✓			↑	↑		
	Graph Analytics	GraphX			✓	↑	↑		
On-line	Data Caching	Memcached	✓	✓			↑		↑
	Data Serving	Cassandra	✓	✓			↑		↑
	Media Streaming	Nginx	✓	✓				↑	↑
	Web Search	Solr, PageRank	✓	✓		↑		↑	↑
	Web Serving	Nginx-Memcached-MySQL	✓	✓			↑	↑	↑

Figure 29 Classification of cloud applications and typical frameworks employed

In VINEYARD, we plan to target specifically the acceleration of cloud applications that are mainly CPU-intensive and that can benefit most from the utilization of the hardware accelerators.

8.2.1 Throughput

The main metric for the evaluation of cloud-computing application is throughput. Different metrics are used to measure the throughput based on the application characteristics:

- The throughput can be measured as the number of processed ***requests per second*** (RPS, in short) for online service workloads[32].
- The number of ***operations per second*** (OPS, in short) is used to evaluate OLTP workloads (OLTP refers to Online Transaction Processing).
- For data-analytics workloads, the throughput is measured as ***data processed per second*** (DPS, in short). DPS is defined as the input-data size divided by the total processing time. In comparison with metrics like the processed jobs or tasks per time unit, DPS is much more relevant to the data processing capability of the system which users concern.

[32] Lei Wang et al. BigDataBench: a Big Data Benchmark Suite from Internet Services, 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)

8.2.2 Latency

Another major metric in the cloud-computing applications is latency. Latency refers to the overall time it takes for the completion of a specific task/job. The latency is especially crucial to applications that have to do with human interaction like *web search*. The typical metric is the average latency for the completion of the tasks. However, average latencies only give half the story. A more important metric is the *N-th percentile latency*.

N-th percentile latency refers to the percentage of measurements that experience a specific threshold latency and above. If this percentage is high, then the system experiences Longtail latencies. ***Longtail latencies*** occur when high percentiles begin to have values that go well beyond the average and can be magnitudes greater than the average. A 99th percentile latency of 30 ms means that every 1 in 100 requests experience 30 ms of delay. For a high traffic website, this could mean that for a page with 1 million page views per day then 10,000 of those page views experience the significant delay that may affect the operations of the web page[33].

The most typical metrics used for the N-th percentile latency are: 90th% latency, 95th% latency and 99th% latency. For example, the requirement for the memcached applications is 95th % latency of less than 300 μsec, as is shown in Figure 30 and Figure 31, below.

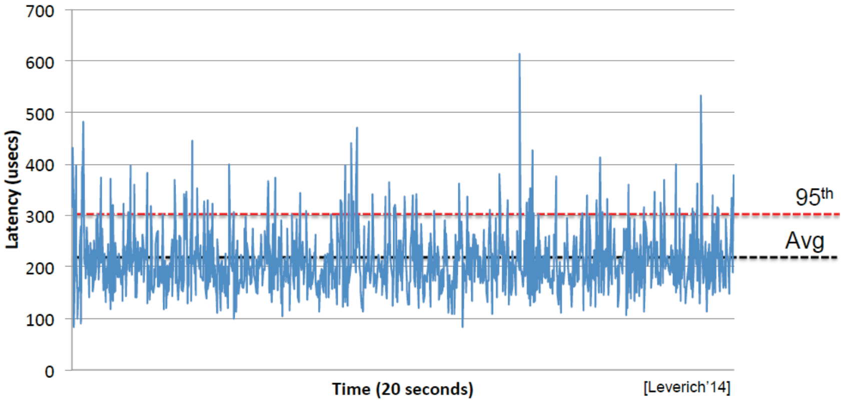


Figure 30 95th-percentile latency for memcached [Source: Kozyrakis[34]]

[33] Who moved by 99-th percentile latency, https://engineering.linkedin.com/performance/who-moved-my-99th-percentile-latency

[34] Christos Kozyrakis, Resource Efficient Cloud Computing, IAP cloud computing workshop, 2013

European Commission
Co-funded by the Horizon 2020 Framework Programme of the European Union under Grant Agreement nº 687628

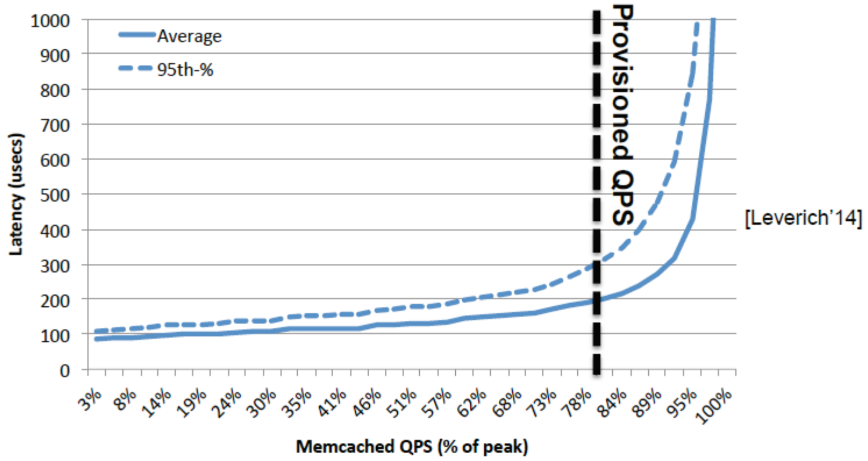


Figure 31 Queries-per-Second (QPS) versus Latency [Source: Kozyrakis[34]]

8.2.3 Completion time (Total execution time)

The Completion time (or total execution time) refers to the overall execution time of the applications. This is mainly used in applications like batch processing where the application need to process a large file (i.e. Big Data analytics). The execution time takes into account both the application code and the operating system time. To avoid interference with other applications, the total execution time of a specific application should be measured when only this application is running on the system. Alternatively, if we care about the total execution time of a mix of applications (e.g. a benchmark with data analytics, in-memory analytics, web serving, etc.) then we can measure the total execution time to finish all the applications in the system.

8.3 Motivation for application acceleration

Spark is a programming framework that allows the fast implementation of big data analytics applications in large distributed computing systems. Based on the Spark libraries, different types of applications can be developed such as machine-learning, computational graphs and database applications. Most of these applications such as machine learning are very computationally intensive and much higher performance is required by the applications users.

8.4 Tasks suitable for acceleration

In VINEYARD we will first explore if there are specific tasks in the Spark's kernel that could be offloaded to the accelerators. For example, Spark allows the compression of the datasets that are distributed to the nodes. The accelerators could be utilized in this case to offload the processor from the computationally-intensive tasks of compression.

During workload characterization, we will also study the tasks of widely used applications based on Spark and identify the most CPU-intensive tasks that could be accelerators. For example, there are several machine learning applications based on Spark such as logistic regression, K-means clustering, Support Vector Machines and Naïve Bayes with computational intensive tasks that could offloaded to the accelerators.

8.5 Summary

Besides the three use-case scenarios from the applications partners, VINEYARD will also study how widely-used programming frameworks for Big Data analytics, such as Spark, can benefit the most from the efficient and transparent utilization of the hardware-accelerator-based servers that will be developed in VINEYARD. Specifically, VINEYARD aims to provide an integrated framework that will allow the speedup of the Spark applications and the reduction of the energy-consumption by the efficient utilization of the accelerators.

Most of the Spark applications that will be examined are batch processing applications (e.g. machine learning applications). Therefore, there is not any strict requirement in terms of latency or throughput. The main goal in this domain will be the speedup of the completion time by at least an order of magnitude and the reduction of the energy consumption at least by 20x.