MIKELANGELO

D2.1

First Cancellous bone simulation Use Case Implementation strategy

Workpackage:	2	Use Case & Architecture Analysis
Author(s):	Uwe Schilling	HLRS
	Nico Struckmann	HLRS
	Michael Gienger	HLRS
Reviewer	Matej Andrejašič	Pipistrel
Reviewer	Nadav Har'El	Cloudius
Dissemination Level	Public	

Date	Author	Comments	Version	Status
2015-08-07	Uwe	First notes / Initial draft	V0.0	Draft
2015-08-13	Nico / Uwe	Enhancements and refinements	V0.1	Draft
2015-08-14	Uwe / Michael	Document ready for review	V0.2	Review
2015-08-31	Uwe / Michael	Document ready for submission	V1.0	Final

Executive Summary

This is the first of three deliverables regarding cancellous bone simulation. Each deliverable will contain the progress done and will describe the next steps. The cancellous bone simulation is one of the use cases of MIKELANGELO and is part of the work package 2. This document describes the bones software and tries to give some insight how this software works and what it needs to run. It explains limitation that are given at this point in time and explain how the MIKELANGELO project can improve the current work-flow in HPC centers. Furthermore, an implementation plan is outlined as well as an evaluation plan to document the progress and compare the results.

Table of Contents

Table of Figures

Glossary

CPU	Central Processing Unit
CT	Computer Tomography
FE	Finite Element
GB	Gigabyte
GB/s	Gigabyte per second
Gb	Gigabit
Gb-Lan	Gigabit-Ethernet
Gb/s	Gigabit per second
HPC	High performance Computing
I/O	Input / Output
InfiniBand	computer-networking communications standard
KPI	Key Performance Indicator
KVM	Kernel-based Virtual Machine
LTS	Long Term Support
LTS	Long Term Support
MB	mega byte
MPI	Message Passing Interface
MRI	Magnetic Resonance Imaging system
NFS	Network File System
OS	Operating System
OSv	The new develop Operating System
OpenMP	Opem Multi-Processing
Qemu	Quick Emulator
RAM	Random Access Memory
RDMA	Remote Direct Memory Access
RVE	Representative Volume Element
Req #	Requirement number
RoCE	RDMA over Converged Ethernet
sKVM	extended hypervisor
VM	Virtual Machine
Xen	hypervisor which uses a micro-kernel design

1 Introduction

This is the first of three deliverable's regarding Cancellous bone simulation. Each deliverable will contain the progress done and will describe the next steps. The Cancellous bone simulation is one of the use cases of MIKELANGELO[1] and is part of work package 2 (Use Case & Architecture Analysis). It is a high performance application software that is developed for HPC centers like the High Performance Computing Center Stuttgart (HLRS) and can run on several thousands of compute nodes and scales accordingly. It calculates the density of the cancellous bone (also known as trabecular or spongy bones) to develop more accurate and longer lasting implants to replace damaged human body parts.

The development process of such a highly optimized HPC application software is one of the current challenges to be focused on by the MIKELANGELO project. New efficient software for supercomputers and HPC clusters are challenging to develop and demand deep insight knowledge of the software and hardware used in high performance computing (HPC) centers. The MIKELANGELO project addresses this problem with the help of abstraction layers to simplify HPC application development, by designing a new hypervisor and a lightweight guest operating system. Also, it tries to optimize the work-flow of HPC centers for the provisioning of new applications, especially if they have conflicting requirements regarding the current set up (e.g. particular kernel version is required) and try to simplify the deployment of such software. To improve this new software stack it is necessary to measure imported parts (node communication / I/O) of the software stack. The key performance indicators (KPI) that we like to measure are performance oriented, due to our businesses model. These KPIs are guiding the implementation of the new software development within MIKELANGELO. HLRS provides a hardware test platform which is configured with similar hardware and software used in a production system to show the integration steps necessary to implement the MIKELANGELO software stack into a HPC center.

[1] http://www.mikelangelo-project.eu/

2 Main Deliverable Content

Biomechanical research relies increasingly on simulations to develop more accurate and longer lasting implants to replace damaged human body parts. The positioning and geometry of these implants can be vastly improved by knowledge of the bone density which can be precisely calculated with the help of HLRS' cancellous bone software.

Typically, Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) are techniques to gather data for such a medical analysis. However, in the case of the bone structure, with various density, this information can be refined further with precise computer simulations of the bone's structure. To simulate the micro-structure of a cancellous bone (also known as trabecular or spongy bones) there are several steps necessary. The general micro-structures are approximately 0.1mm in diameter, only, but to be able to distinguish these structures precisely, an additional processing step is required to achieve a refinement of the resolution down to 0.02mm.

2.1 Use Case Description

The cancellous bone simulation is an I/O intensive application. To calculate the structure of the bone it has to split up the data set into smaller pieces. The division of the whole data set (derived from CT / MRI scan) into tiny cubes, which can be processed by the workers, is handled by the master process. These data sets are stored by the master process in a queue from which the workers fetch their input for the calculation of a bone's structure.

Depending on the actual structure of a data-set, which represents a tiny cube (which is called RVE, Representative Volume Element) of the whole bone structure, the calculation time strongly differs. This time cannot be assessed reliably in advance.

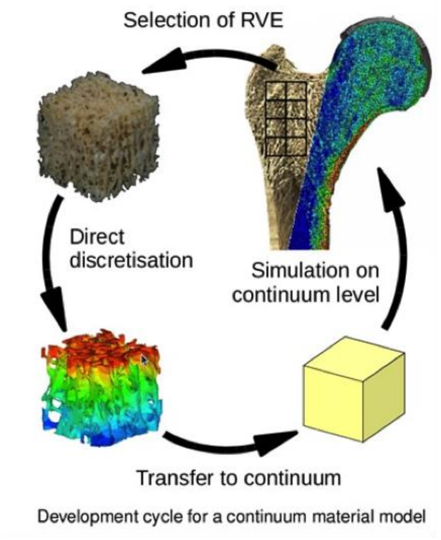
Selection of RVE
Direct discretisation
Simulation on continuum level
Transfer to continuum
Development cycle for a continuum material model

Figure 1: Data workflow of the cancellous bone application

Therefore, the data set is called asynchronous, since it is not possible to split and distribute all cubes to all calculation processes beforehand evenly regarding their execution time. It has to be done while the application is running. The worker will fetch the next part of the calculation after it has finished its prior work. The master will provide the parts and orchestrates the work flow. To calculate the structure of a bone sample, 27 Finite Element (FE) calculations are necessity. This data is stored as intermediate result during the calculation.

To keep all workers under load it is necessary to serve them data sets with low latency and high throughput (see 3.8 Initial Measurements), since the waiting time of the worker for retrieval of the next data set cannot be used otherwise. If there are parts of the cube with less calculation time it is possible that workers fetch data sets in tiny intervals. Pre-fetching data would help to save maybe milliseconds of overall calculation time, however, this is a lot of work and based on guesses which might be completely wrong - the most efficient approach is the master-worker scenario. Thus, a low latency is beneficial as well as a high throughput is if there are many workers fetching data in parallel.

2.2 Current Limitations

Due to optimised compilers for different architectures (e.g. Cray Compiler, Intel Compiler, Gnu Compiler) the simulation software requires a rebuild on different hardware architectures. It needs to be slightly modified each time to match paths for libraries or particular versions of a library. This slows down the development and build process of this kind of HPC software massively, if is intended to run on any other hardware than the development environment consists of. Furthermore, it is not possible for users to install particular kernel on the cluster hardware, so they have to use the installed software if there is a hard restriction on a specific library version they have no possibility to run the software.

As the Bones application is clearly a high performance application, the overall setup of the software stack is complicated and requires various standard packages as well as self-compiled extensions and libraries. Thus, a clear benefit for the overall bones setup phase would be the ability to install required software components automatically by defining the packages and archives including their particular versions. As the current process foresees human interaction and manual configuration, an automated mechanism to provide the execution environment would ease the whole execution process. Alternatively, a complete software stack could be offered directly based on pre-configured images ready to be run.

2.2.1 HPC

In the field of high performance computing, the biggest problem is scalability, beside this there are obviously no clear limitations in the sense of performance. The application scales very well and has already been used in huge-scale dimensions, in particular with 32.000 cores on the HLRS' Cray supercomputer Hermit[2].

However, the HPC clusters are highly specialized and thus, clearly lack on service provisioning and flexibility. At first, a signed contract between the HPC centre and the user is

[2] https://www.hlrs.de/systems/platforms/retiredsystems/crayxe6hermit/

necessary before getting access to any system. Most parts of the system are not user friendly. To prevent performance bottlenecks users have to live with some inconveniences, e.g. graphical user interfaces are avoided. Hence, access to the systems is only granted via remote shell and highly secure connections using the standard protocols and certificates (X509, GSSAPI, etc). Furthermore, due to the specialization, special requirements and parameters have to be fit in order to achieve high performance. When executing the applications: for instance, compiler options and flags to tune the application in the best way have to be known and set afterwards.

2.2.2 Cloud

In cloud systems, the statements above are basically contrary: the service and usage itself are well defined and easy to access, but the performance is far from specialized high performance computers. In clouds, I/O is designed for providing a standard user service, for example a multi-tier web application. The requirements for very low latencies and high bandwidth are mostly not met and thus, represent a major bottleneck when executing high performance applications. In addition, this also counts for the network interconnects. In HPC, mostly InfiniBand and proprietary solutions are used, which provide ultra-low latencies, high throughput and remote direct memory access (RDMA) that speeds up processing and communication amongst workers remarkably. Another issue caused by the fact, that the virtual instances have to be started before the actual application can be executed. This amount of instantiation time is significant for a huge amount of instances. Finally, the overhead for full virtualization with Xen[3] or KVM[4] eliminates the advantage of cloud computing for high performance applications.

These disadvantages also apply for the introduced high performance application use case Cancellous Bones simulation. It requires a very fast storage back-end and interconnects capable of RDMA and high bandwidths with low latencies between the nodes that traditional clouds cannot provide. Even if there would be the possibility to have for example InfiniBand, there is still the huge overhead of the operating system that slows down the applications and in parallel, increases the execution costs.

2.3 Expectations for the MIKELANGELO Stack

MIKELANGELO focuses on the simplification of HPC software development and HPC application execution. This will be achieved by simple work flows. E.g. the user can fetch a standard VM from our stack, build his software in a pre-defined environment without having access to HPC hardware (i.e. on a desktop computer) and then push the prepared VM including his developed software to any cluster offering hypervisor(s) for the execution of virtual machine images and define how it should run.

Currently, developers have to learn in great detail on the individual setups of the system, to adapt their software and to achieve maximum performance. This includes the overall structure

[3] http://www.xenproject.org/
[4] http://www.linuxkvm.org/page/Main_Page

of the system and available software components. They have to modify the software in order to run in a particular HPC environment.

An improved start-up time would be beneficial for the workload of the whole compute environment. To get closer to a bare metal HPC execution, it is essential to reduce any kind of overhead to an absolute required minimum. This does not only affect booting times, but also the latencies and bandwidths for all kinds of executions.

Especially the performance for I/O and network is a crucial requirement at the current point of time as this limits current cloud computing performance. Thus, MIKELANGELO is expected to improve this circumstance significantly. As the Bones application, besides all other pure High performance applications requires a shared storage for all nodes to store intermediate as well as final results, the performance of this component needs to be optimized in terms performance.

The following bullet points summarize our expectations regarding achievements of the MIKELANGELO project:

- Improved start-up time for virtual resources
- Support standard HPC hardware (x86 CPU architecture), like storage systems and interconnects
- Support HPC software mechanisms and protocols, like RDMA or RoCE as well as super down-striped operating system
- Minimise the performance overhead of full virtualization
- Enable easy orchestration and management by automation and contextualization

3 Use Case Set-up

The Bones Use Case is a good example of an HPC application. It has the same restrictions (IO-bound) as many other HPC software and provides solid and reliable results. The main intention of this use case is to measure how much performance is lost due to the virtualization - at this point, with the latest versions of KVM and a standard Linux based OS. The expected drops are in the plain run time, due to the start-up time of the virtual machine and in the network latency and the IO. This performance decrease is expected, each layer of software will add an overhead and decrease the performance.

To validate the improvements resulting from developments within the MIKELANGELO project, KVM will be swapped to the newly developed sKVM and the standard Linux based guest OS will be replaced by the lightweight single user cloud operating system OSv[5]. In each step the use case will run on the same hardware (see 3.1 Physical Hardware) to get comparable results. We will run each test multiple times to minimize measurement errors. The expectation for MIKELANGELO is, that with each step forward of the MIKELANGELO project, the performance will be increased and the overhead for virtualization decreased down to a point where it is feasible for deployment in HPC production environments.

3.1 Physical Hardware

The focus of our hardware setup is to be as close as possible to HLRS' production environment. The Blade-center dedicated to the MIKELANGELO project is inter-connected via Gigabit Ethernet as well as InfiniBand. The InfiniBand is a network interface which provides a low latency network with and a 1.2 µs MPI ping[6] time combined with high bandwidth of 10Gb/s and is capable of RDMA.

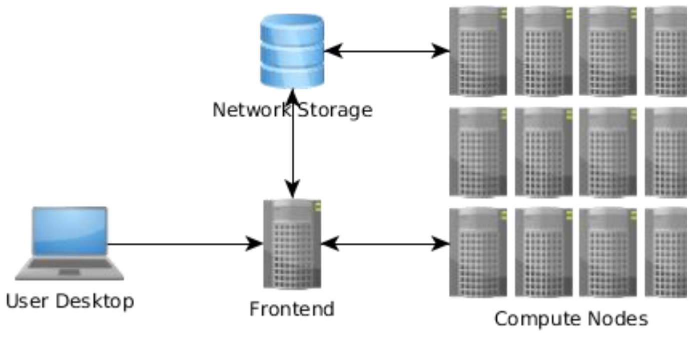

Figure 2: Testing environment set up

[5] http://osv.io/
[6] http://www.mellanox.com/page/products_dyn?product_family=4

The user $HOME directories are shared across all nodes (NFS) offering a shared and fast network based file-system for the data exchange during calculation and intermediate results. The NFS storage is hosted on a dedicated server. Project partners can access it through a publicly available front-end where they are able to submit batch-jobs with the help of the resource manager Torque.

The Blade-center consists of 14 nodes with 32GB RAM and 8 cores each. This adds up to a total of 112 cores and 448GB RAM. The dedicated front-end node offers 4 Cores and 16GB RAM.

3.2 Software

All 14 nodes have the same operating system installed and software packages are also the same on all nodes. The exception is one node which provides a network manager for the InfiniBand network, on this node are additional packages installed for software building and administrative work. The cancellous bone application should run on all 14 nodes with all 112 cores at the same time. It will use the InfiniBand interconnect to communicate between compute nodes and will read from and write to a shared file system storage (NFS[7]).

The cancellous bone application is dependent on the following libraries:

- libcr-dev, a library to checkpoint/Restart programs. Allows programs to be check pointed, stopped and later restarted
- mpich2, high performance implementation of MPI (Message Passing Interface)
- gfortran, implementation of the GNU Fortran compiler for gcc
- libmetis-dev, libmetis5, math library for fill-reducing matrix ordering and serial graph partitioning
- fmps framework for finite elements
- gcc, the GNU Compiler Collection
- petsc[8], a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.

7 http://nfs.sourceforge.net/
8 http://www.mcs.anl.gov/petsc/

3.3 Execution

In the following subsections, the execution of the Bones application will be highlighted. Therefore, two sections have been chosen to discuss the application execution as well as its research and business focus.

3.3.1 Application Execution

The cancellous bone application has three phases in its life cycle. The first phase will split up the whole data set in smaller domains (tiny cubes). This part is very fast due to the data format PureDat described in detail in the subsequent section (see Section 3.4 Data).

The second phase is the compute intensive part. At this point a master process keeps track over the calculated domains, the worker collects the domain information and fetches the domain's specific data set from the shared storage (NFS) and starts the calculation. The calculation time depends strongly on the domain size and the actual bones structure within a domain. Cancellous bone have denser and less dense parts. Denser parts are compute intensive and need longer calculation time. In contrast, hollow parts can be calculated nearly instantaneously. The results are written to the NFS and will be combined in phase three. Each time a worker has finished its calculation it fetches the next part form the data queue (MPI[9] communication). Hence, each worker should spend the most time for calculation. This strongly depends on the speed of data delivery towards the worker.

The third phase is the combination of the calculated subdomain data into a global picture. This is a complex part from the mathematical point of view, but the calculations are fast in comparison to the overall run time.

Execution time (on standard laptop for a single RVE):

- ~ 5 minutes (small development test)
- ~ 45 minutes (bigger test)
- ~ 24 Hours (data set with high resolution)

3.3.2 Research and Business Focus

In contrast to the subsection above, this section details the research and business focus of the use case. As already detailed, the Bones application has been already executed on state of the art supercomputers with a huge amount of computing cores. However, this mechanism is not very practical for the foreseen end users due to the limitations of HPC. For this purpose, distributed executions have been foreseen in order to enable efficient and in-time processing.

This business case directly bundled to the Bones application are hospitals and bone implant producers that need to improve the quality of service for patients, their direct customers. So far, only a few different sizes of implants for hips, for instance are available. In conclusion, this leads to a huge discrepancy for people who don't fit those standardized sizes. To

9 http://www.mcs.anl.gov/research/projects/mpi/

accomplish this ambitious goal, we foresee the usage of all available personal computers and laptops to process during their idle times the required calculations. Using a process like this, the time between diagnosis using modern computer tomography, the selection of the best suitable implant and its production can be significantly improved. As the implants are produced on the basis of the patient data, a significant extension of the usage from currently approximately 10 years is expected. In addition to this, the sickness absence rate will be drastically reduced based on the optimized implants, as well fitting implants will improve and shorten rehabilitation.

In order to achieve the business goals by enabling a distributed and scalable service, there are different technical issues that have to be resolved. On the one hand, a distributed version of the application needs to be created. On the other, its execution parameters and performance have to be understood in detail. At this point, research is still required.

Within MIKELANGELO, the application will be executed in a distributed manner. The simulation results of the application are not of importance, it is rather important to understand its execution parameters in detail.

For instance, what are the concrete differences between the following two examples:

1. Co-location of virtual instances

In cloud environments, resources are always shared by the co-located virtual machines. This use case will examine the parameters for this co-location. Is it beneficial for the execution that VMs are located at the same physical host so that RDMA protocols work highly efficient? Or does resource sharing of networks prohibit high performance? Those questions will be resolved within the Bones use case.

2. Virtual instance size

The size of virtual machines is a critical factor for the application as well. So far the implications between core sizes of virtual instances and their memory are unclear. In other words, are there parts within the application that run efficient with a small amount of cores or memory, which would translate to a notebook within a hospital or are bigger VMs with higher performance required? As above, questions like these are of high interest and need to be clarified.

As can be seen, although the business aspect is targeted, research is still very important for the Bones use case. Therefore, various kinds of executions with different parameters for CPU, memory, network and physical co-location will be used in order to evaluate the distributed performance of the use case.

3.4 Data

The file format for the bones application is called PureDat. The fundamental idea behind PureDat was to develop a file format which separates the data types in memory to minimize the necessary system and library calls. PureDat is used as input file format as well as output. The data set is split into 4 files with a total size of about 19 GB for a medium sized set (maximum data set is ~40 GB). One of the files is the main data set and contains most of the data used. The data for the calculation will be provided via a NFS (Network File System) which is mounted on every compute node as well as the frontend. This NFS-Server is an independent machine and is only used as storage server. The nodes itself will communicate over MPI or the filesystem, to collect the data for calculation from the NFS. The master process will mark the domains and send the information (where they located) to the worker. The worker collected then the data from the NFS server. As mentioned in the previous parts of this document, the main problem for HPC application is to get the data as fast as possible to all nodes and provide a fast storage to perform at the maximum capability's of each node. Nodes without data can not calculate. To minimise the latency and maximise the network throughput, the communication (MPI) runs over InfiniBand and data transfer is handled by Gb-Lan.

3.5 Security

The sample data set for testing as well as for the performance measurement are trouble-free regarding security and data privacy. Real data sets however are very critical regarding data privacy and need therefore to be protected strongly against any kind of data leakage.

3.6 Mandatory Requirements

Almost all simulations depend on some sort of shared workspace. This is uses to load and save intermediate data accessible for all nodes while the application is running. For high performance applications it is necessary that this storage is as fast as possible. While most nodes make use of their local storage as temporary cache, only. The standard for applications to communicate in a HPC environment is MPI and OpenMP[10]. This is also needed for the cancellous bone simulation.

Following requirements (see D2.19 The first MIKELANGELO architecture for a full list) have been identified for the cancellous bone application:

- **Req #038: Bones app must be running in OSv and sKVM**

In order to execute the code within the OSv operating system, various changes for the Bones application have to be targeted. Especially all the components need to be compiled as shared objects, which will impose a complete restructuring of the application.

- **Req #104: Integrate NFS Client in Osv**

[10] http://openmp.org/wp/

For the execution of the use case, a shared workspace is mandatory, as detailed above. HPC environments use Lustre file systems for this purpose, for Clouds a network file system will be enough. For initial executions, the NFS client will be required as the server will be hosted on a dedicated physical machine. However, the final executions foresee a full cloudified solution, including a NFS server.

- **Req #105: NFS Server should be available in OSv images**

As detailed above, the final executions foresee a full virtualization of the Bones software stack. Thus, a NFS server is required as well. However, the application runs without a dedicated server, so this requirement can be marked “optional”.

- **Req #077: support for InfiniBand core driver**

In order to speed up MPI communication in terms of latency and bandwidth, InfiniBand network support is recommended for this application. Therefore, the hypervisor as well as the operating system need to support this kind of interface.

- **Req #027: OSv support for message passing (MPI)**

The Bones application uses the message passing interface for the communication mechanisms between the processes and threads. This library has to be ported to OSv in order to guarantee efficient inter process communication.

- **Req #025: OSv support for InfiniBand hardware (virtual interfaces)**

This requirement is tightly coupled with req #077. In order to establish the InfiniBand connection, both requirements have to be fulfilled.

- **Req #030: OSv support for RDMA (core driver of InfiniBand)** RDMA (Remote Direct Memory Access) is part of the InfiniBand functionality and increases the cross CPU socket/ cross node communication drastically. RDMA is one of the core principles for high performance computing. Therefore, in order to enable high performance clouds, this mechanism needs to be available as well. The Bones application will make use of this kind of functionality using InfiniBand and MPI.

- **Req #042: Capture performance metrics of guest OS – Osv**

To understand the performance of the operating system and with this, the application in detail, it needs to be possible to capture monitoring information dedicated to the application execution.

- **Req #041: Capture performance metrics of host Hypervisor – sKVM**

In order to compare the results of bare metal and virtual executions, the performance difference has to be assessed. This performance metrics define the entire result of MIKELANGELO and therefore, great attention has to be put on those.

- **Req #10: Hypervisor support for Ubuntu guest**

For the comparison of different guest operating system, at least one other OS needs to be supported by the hypervisor.

Besides the aforementioned requirements, there are other various related requirements, especially for the network and protocol integration that need to be achieved in order to enable the core functionality of the Bones application. All related requirements can be observed in D2.19 - The first MIKELANGELO architecture.

3.7 KPI

The business model of HPC-Centers is to provide as much performance as possible. Software developers and application users have to accept several limitations to get the best performance out of their code. Therefore, the key performance indicators (KPIs) of this use case are capturing different aspects relevant for the overall performance of the application.

Briefly summarized, we are regarding these metrics:

- plain run time
- metrics from the application
    - time to load data (bandwidth and latency / random read and write)
    - time for calculation (computing efficiency)
    - time to store data (sequential read and write)
- Overhead measurements
    - metrics from OSv / basic VM operating system
        - w/r data
        - network communication
    - metrics from kvm/skvm
        - w/r data
        - network communication

KPIs from Grant Agreement:

[KPI2.1] relative efficiency of virtualized I/O between KVM and sKVM (developed in the project)

To measure the improvements we will collect data from of the application itself. Most of the HPC application measure several things during execution and collects monitoring log file. The interesting things here are the time the application need to load the data (bandwidth and latency combined). As described in the previous sections this is the most essential aspect regarding the use case's performance (see 3.3 Execution). How many data junks that are load and stored while the application is running, is more exactly explained in 3.8 Initial Measurements.

[KPI3.1] The relative improvement of efficiency of MIKELANGELO OSv over the traditional guest OS.

We will also gather information from application, therefore we will compare an Ubuntu VM with the new develop OSv. Start up time is only one factor which will be improved. To increase the efficiency I/O and calculation overhead are other characteristics which are important. To measure these it will be necessary to compare this two approaches similarly. As described in 3. Use Case Set-up we will measure different combination of hypervisors and VM to cover all possible configurations. Regarding the efficiency improvements we are interested in the star up time as well as I/O throughput and CPU calculation time. The last of these three metrics is also part of the hypervisors and will be examined by the cross-level-optimization as well. In detail, the pass-through form VM workers to the hypervisor to bare metal (physical memory and CPU) works. **[KPI3.1, KPI3.3]**.

[KPI3.2] The relative improvement of efficiency [size, speed of execution] between the Baseline Guest OS vs. the MIKELANGELO OSv.

One of these KPIs is the plain run time of a simulation. This includes the VM start-up time, the actual runtime of the application itself and the shutdown of the VM. The overhead of the VM needs to be as low as possible. The faster the VM is ready and shut down after the run the more jobs can run in total. In addition, disc space should be as low as possible to have the most of space for calculation data. Reliable, secure and fast storage is an expense factor and will rise when try to scale up data systems for high performance computing. The size measurement is fairly simple. We compare the actual disc size of the different operating system images (OSv and Ubuntu). For the executing part of this KPI we compare the computing part separately as well. To see how well the different VM will communicate with the underlying hypervisor and the CPU.

[KPI3.3] The relative improvement of compatibility between baseline and MIKELANGELO versions of OSv.

Due to the integration of our use case we will show that OSv is compatible with a state of the art Simulation software and is capable of execute this software accurate. Built in the Cancellous bone application are measurements which indicate the outcome of the simulation and if the calculation are correct. Therefor we can prove that OSv is suitable for HPC applications and will be a powerful alternative to traditional virtualized operating systems.

With the help of these KPIs we are able to compare the application performance and identify bottlenecks affecting a virtualized execution, which needs to be focused on during the project. The last part of the KPI are built around the data pass-through between the software layers. We would like to measure in (s)KVM as well as in OSv how the data generated by the software flows through the several layers. To achieve this we will use monitoring hooks that are built in (s)KVM for network communication as well as read / write performance. This two indicators will be measured in OSv and at the basic VM operating system.

In the end we will have a nearly complete data flow measurement and computation performance thou the different software layers and have comparable results over the different intermediate steps within the MIKELANGELO project. In addition, we can compare the overhead regarding size and overall performance.

3.8 Initial Measurements

The following measurements are taken from execution runs on the HLRS Cray-XE6-System [4]. The granularity of configuration determines how precise the results of the calculation are. If the data set is split into cubes with 0.6 mm pieces, there are 169.344 Representative Volume Element (RVE) for the calculation process. If the initial implementation is used each RVE has a calculation time up to 40 min which leads to an overall calculation time of roughly 8 years on one single core. To speed up the process, a parallelization is mandatory. Different approaches of parallelization had been tested during the development process of this application.

One approach is to synchronize the master and workers over the file system. This approach can scale-up to ~900 nodes (~ 30.000 cores) and will stop scaling in this range of cores. But it is possible to calculate 20.000 RVE in under 12 hours and the whole data set in about 4.25 days. Another approach uses MPI for synchronization and can scale-up even further.

Serial calculation of one RVE:

Process	Calculation time [sec]
geometry extraction and model setup	~ 10
FE-Simulation	~ 1480
Calculation of material properties	~ 10

The mean data throughput of one RVE is ~ 0.054 MB/s. This value is derived from a generated data set of approximately 41 MB. During the execution 40 MB of data has to be read for one RVE, this is then divided by the calculation time of about 25 minutes and the 27 FE-Simulations processed.

The problem at this point is, one node of the XE6 has 32 cores. Therefore, 3200 nodes generate a constant read and write of 40 MB junks with a total bandwidth of ~5.5GB/s.

4 Implementation Plan

HLRS set up a small cluster with 14 nodes. The software that we are using in our production environment to schedule batch jobs is Moab in combination with the resource manager Torque[11]. Torque is open source and offers simple scheduling functionality sufficient for our testing environment while Moab is a sophisticated scheduler requiring a commercial license. The main goal is to be as close to the real HPC production environment as possible. Our integration plan is divided into the set-up and configuration of the test cluster environment and subsequent to this, the execution of the bones simulation to verify that the functionality of the test-bed is as intended. Followed by an execution in virtual machines which is then compared to the bare metal execution to identify performance issues and bottlenecks to be focused on in the hypervisor's development.

We will then try to build the application for OSv and start the validation with a single node at first. To progress further we will then set up the test environment to run multiple OSv instances in parallel and measure their performance. To achieve this we will integrate the hypervisors KVM and sKVM in our environment.

Finally, we will execute the whole use case in the distributed and virtualized environment. At this point, focusing of different execution parameters will be at the forefront. This means, that various executions with varying parameters will be necessary. Those runs will be analyzed and understood in detail. Furthermore, in particular a huge amount of executions will be necessary in order to relativise the possible measurement error. For this reason, only smaller data sets, which have not been defined so far will be used.

11 http://www.adaptivecomputing.com/products/opensource/torque/

5 Evaluation and Validation Plan

To get a virtualized baseline we will measure the cancellous bone application with KVM, Qemu and a basic operating system like Ubuntu 14.04 (LTS)[12]. To analyze the performance of OSv we will use this setup and swap Ubuntu with OSv so we will get a baseline for OSv. To get a baseline measurement of the new sKVM we will swap KVM and Qemu and measure with Ubuntu as well as OSv again. Each measurement will run several times and will be documented to minimise measurement errors. In the end we will end with a baseline, a performance information for OSv and sKVM separated, as well as a combined performance information's.

12 http://www.ubuntu.com/server

6 Key Takeaways

As you have seen the Cancellous bone simulation is a complex HPC software, which is one of the use cases of the MIKELANGELO project. We will show how the newly developed sKVM and OSv can be beneficial for HPC centers. We selected a performance oriented measurement pool and aim to evaluate how MIKELANGELO can help to improve the user's workflow within a HPC center as well as the development workflow for new HPC software.

7 Concluding Remarks

As the project proceeds and the integration workpackage starts and makes progress, this use case will also proceed. For the initial and final measurements the test hardware is necessary to have consistent and comparable results. We will measure the elaborated KPIs and have more in-depth knowledge when we are able to evaluate the first results.

8 References and Applicable Documents

[1] The MIKELANGELO project, http://www.mikelangelo-project.eu/
[2] HLRS XE6Hermit,
https://www.hlrs.de/systems/platforms/retiredsystems/crayxe6hermit/
[3] Xen Project Homepage, http://www.xenproject.org/
[4] KVM Project Homepage, http://www.linuxkvm.org/page/Main_Page
[5] OSv Project Homepage, http://osv.io/
[6] Mellanox ConnectX overview, www.mellanox.com
[7] NFS Implementation, http://nfs.sourceforge.net/
[8] Petsc Project Homepage, http://www.mcs.anl.gov/petsc/
[9] MPI Project Homepage, http://www.mcs.anl.gov/research/projects/mpi/
[10] OpenMP Project Homepage, http://openmp.org/wp/
[11] Tourque Project Homepage,
http://www.adaptivecomputing.com/products/opensource/torque
[12] Ubuntu LTS Server, http://www.ubuntu.com/server