A Diskless Cluster Architecture for Hashing Operations

Spyridon Antakis[1,2], Benne de Weger[1], Padraic Garvey[2]

[1] Techinsche Universiteit of Eindhoven
Department of Mathematics & Computer Science

[2] Intel Shannon, Ireland
Digital Enterprise Group (DEG)

Eindhoven, Netherlands
August 2010

Outline

How the idea came up?

Intensive Hashing Operations

- Digital Forensics
  - ⇒ Hard Disk images
  - ⇒ Independent files
- Cryptanalysis
  - ⇒ Rainbow tables
  - ⇒ Brute-force attacks

Public Key Cryptography

- Security Protocols
  - ⇒ TLS/SSL: Web Servers (connections)
  - ⇒ IPSec: VPNs (throughput)

Hardware Accelerators (1)

Various Trends

- Central Processor Units (CPUs)

  $\Rightarrow$ Embedded technology

- Co-Processors

  $\Rightarrow$ Independent PCI cards

- Graphics Processor Units (GPUs)

  $\Rightarrow$ Extremely powerful

  $\Rightarrow$ Parallelization benefits

Hardware Accelerators (2)

CPUs vs GPUs

- It depends on the nature of the task!

  $\Rightarrow$ 75-150 Watt vs 110 -300 Watt.

  $\Rightarrow$ GPUs: 5-15 times faster.

  $\Rightarrow$ CPUs: Support for any application.

  $\Rightarrow$ GPUs: Low-level programming is necessary.

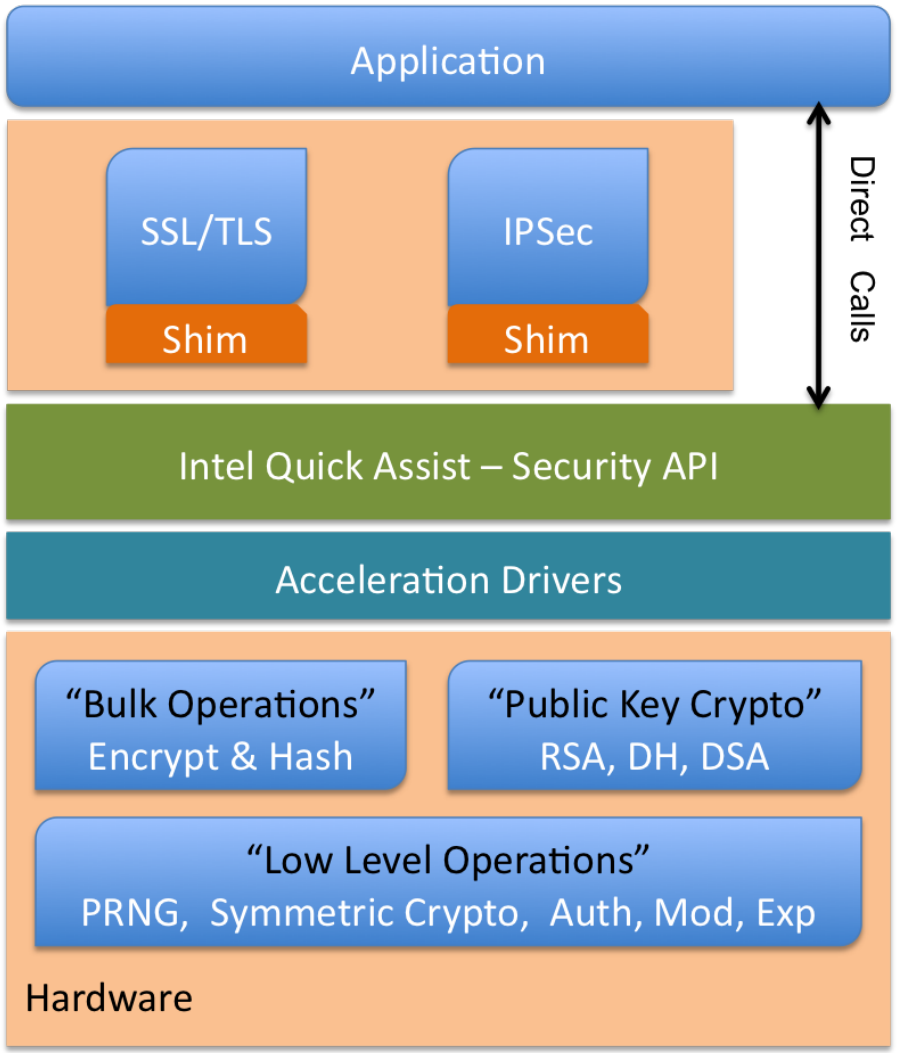
  $\Rightarrow$ Trade-offs: *flexibility*, *performance* & *cost*.

Intel EP80579 Embedded Processor

Main features

- Based on an Intel Pentium M processor.
- Integrates embedded technology for acceleration.
- Supports the most popular cryptographic algorithms.
- High-Performance at low power consumption.
- Released in 2008, the new generation at the end of 2010.
- Dedicated API for acceleration, called QuickAssist.

Intel EP80579 - Enabling Acceleration



Structure & Intentions

Initially

⇒ Use the acceleration of Intel EP80579 (*aka Tolapai*).

⇒ Build a diskless cluster with Intel EP80579 processors.

⇒ Parallelize the tasks into the cluster nodes.

⇒ Investigate the performance of the diskless architecture.

⇒ Benchmark several cryptographic tasks.

⇒ Evaluate the performance of the diskless model.

⇒ Argue about the benefits obtained from the acceleration.

Structure & Intentions

What we accomplished

⇒We built a 4-node diskless cluster based on Intel EP80579.

⇒ We did not enable acceleration (*Intel Pentium M* was used).

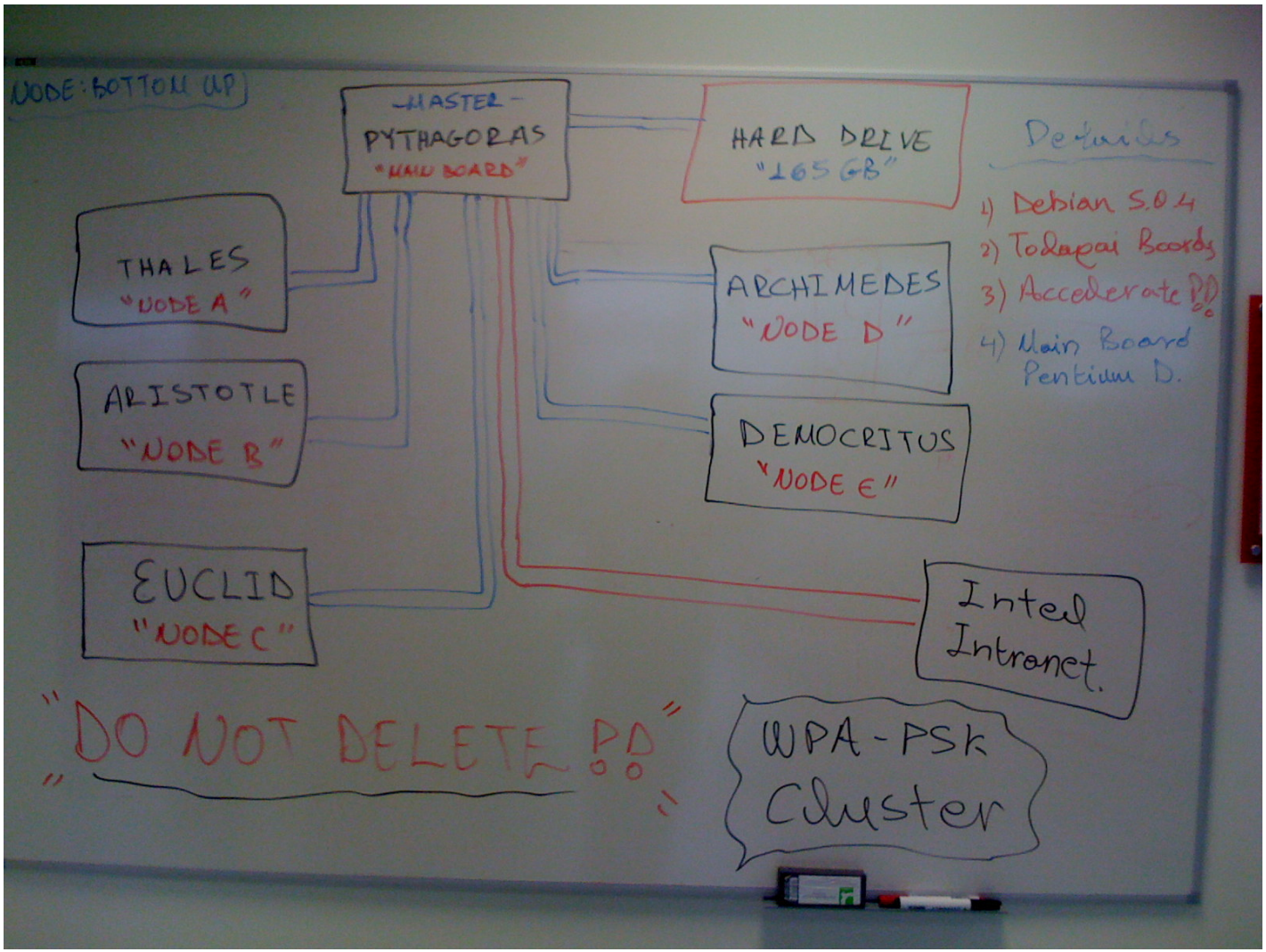
⇒ We tested Hashing Implementations + Skein Algorithm.

⇒ Independent systems were used, as a comparison metric.

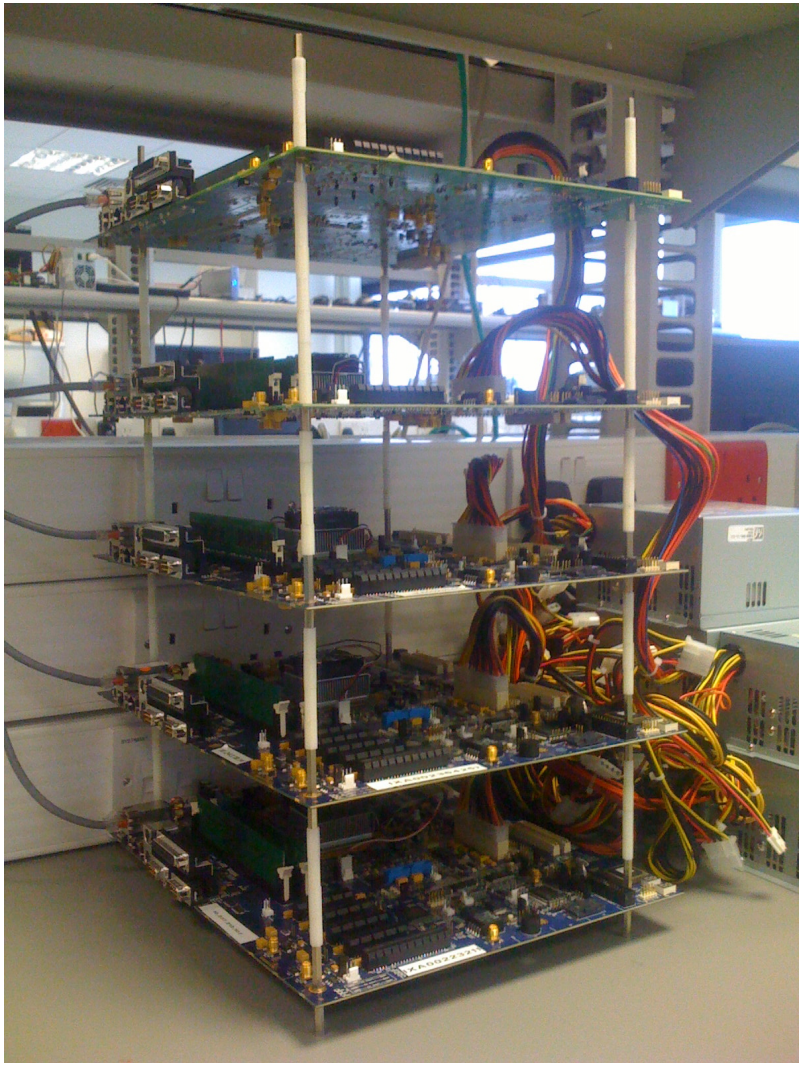
⇒ We parallelized the processing.

⇒ We assessed the performance of the diskless model, based on 2 test cases and we identified the major challenges.

How it started?



How it finished?



What happened in between?

Step 1 : Establish all Physical Connections

Step 2 : Diskless Remote Boot in Linux (DRBL)

Step 3 : TORQUE – "Resource Manager"

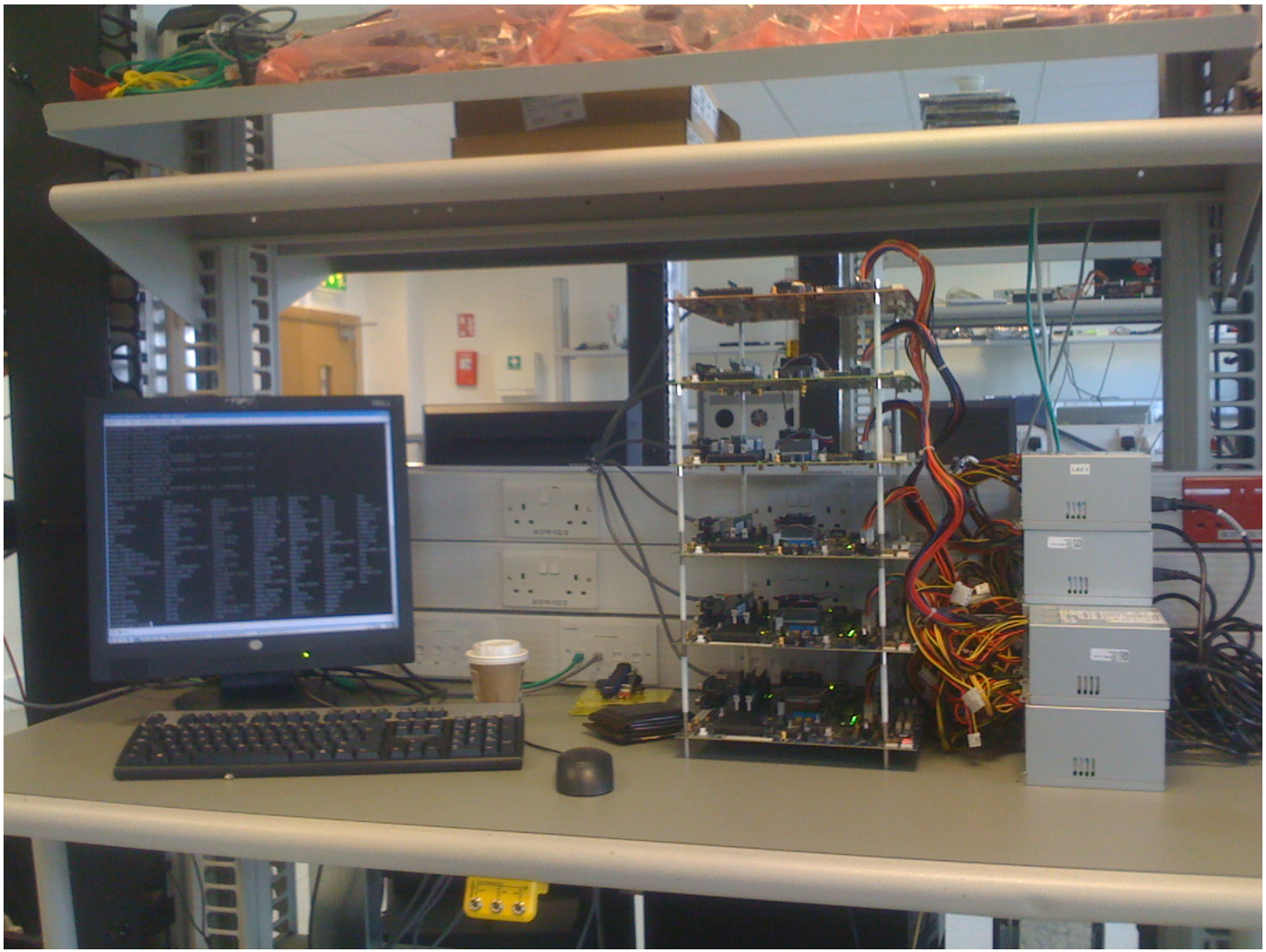
Step 4 : Maui – "Cluster Scheduler"

Step 5 : Install Hashing Implementations

Step 6 : Writing the Scripts

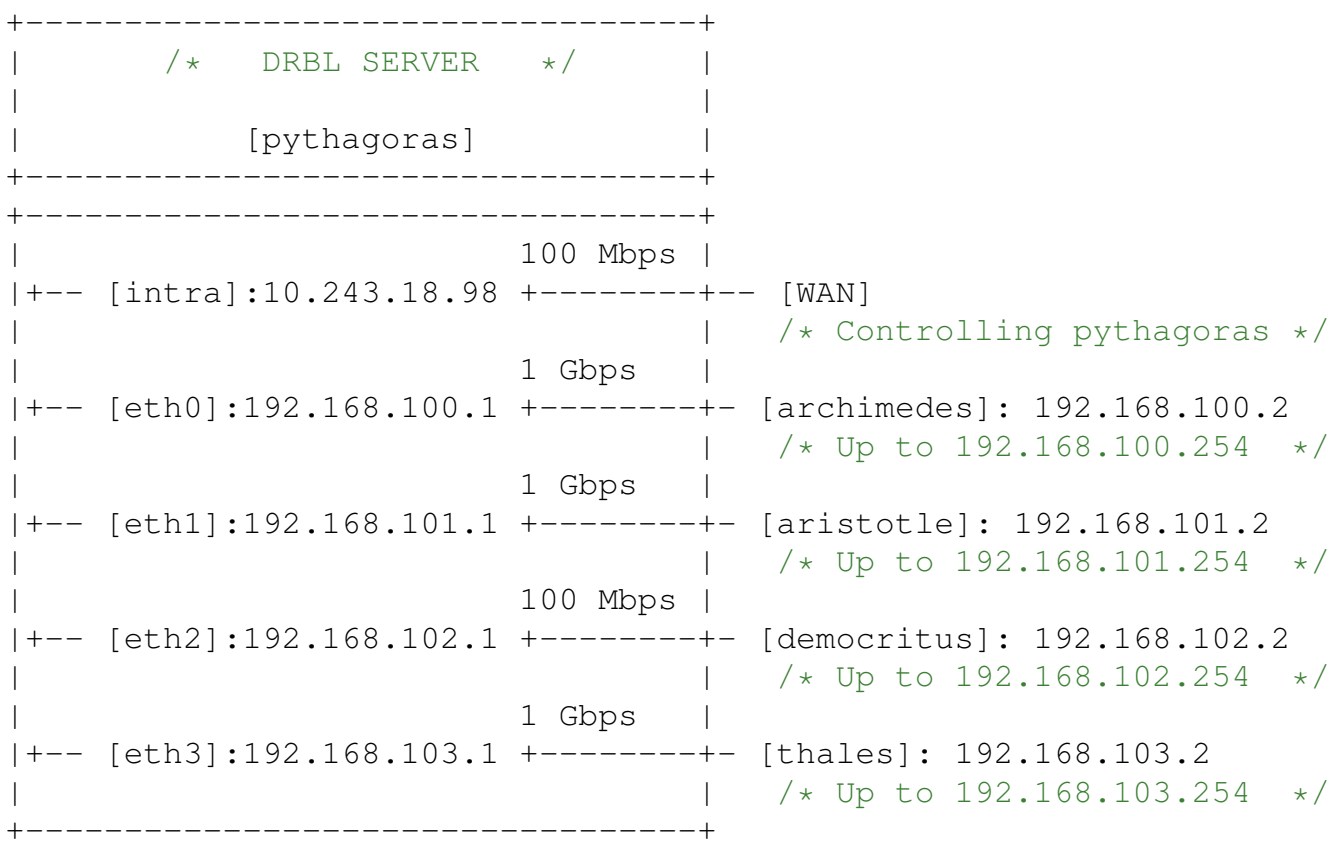
Step 7 : Benchmark

Step 1: Establish all physical connections



Step 2: Diskless Remote Boot in Linux (DRBL)

- DRBL constitutes the main core of the system and it is the software that handles all the communication between each node and the main board.



Step 3: TORQUE - "Resource Manager"

- TORQUE is the resource manager that successfully co-operates with DRBL and gives the ability to schedule, monitor and easily parallelize the tasks into the nodes.

```
[spyros@pythagoras ~]$ pbsnodes -a

node_aristotle02
     state = free, np = 1, ntype = cluster, status = opsys=linux,
     uname=Linux node_aristotle02 2.6.18-8.el5 i686, idletime=4495861,
     totmem=1554188kb, availmem=1483948kb, physmem=1554188kb, ncpus=1,

node_thales02
     state = free, np = 1, ntype = cluster, status = opsys=linux,
     uname=Linux node_thales02 2.6.18-8.el5 i686, idletime=4495893,
     totmem=773912kb, availmem=706124kb, physmem=773912kb, ncpus=1,

node_archimedes02
     state = free, np = 1, ntype = cluster, status = opsys=linux,
     uname=Linux node_archimedes02 2.6.18-8.el5 i686, idletime=4495917,
     totmem=1489156kb, availmem=1419244kb, physmem=1489156kb, ncpus=1,

node_democritus02
     state = free, np = 1, ntype = cluster, status = opsys=linux,
     uname=Linux node_democritus02 2.6.18-8.el5 i686, idletime=4495874,
     totmem=1294100kb, availmem=1224232kb, physmem=1294100kb, ncpus=1
```

Step 3: TORQUE - "Resource Manager"

- The following figure demonstrates an example of TORQUE's **qstat** command.

```
[spyros@pythagoras ~]$ qstat

Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
354.pythagoras            md5sum           spyros          00:00:31 C batch
355.pythagoras            sha256sum        spyros          00:00:34 C batch
356.pythagoras            sha256deep       spyros          00:00:35 C batch
357.pythagoras            sha224sum        spyros          00:00:35 C batch
358.pythagoras            sha512sum        spyros                 0 R batch
359.pythagoras            sha384sum        spyros                 0 R batch
360.pythagoras            sha1sum          spyros                 0 R batch
361.pythagoras            sha1deep         spyros                 0 R batch
362.pythagoras            tigerdeep        spyros                 0 Q batch
363.pythagoras            whirlpooldeep    spyros                 0 Q batch
364.pythagoras            skein256sum      spyros                 0 Q batch
365.pythagoras            skein512sum      spyros                 0 Q batch
366.pythagoras            skein1024sum     spyros                 0 Q batch
```

Step 4: Maui - "Cluster Scheduler"

- Maui is an additional scheduler that co-operates with TORQUE, providing advanced features.

```
[spyros@pythagoras ~]$ showq

ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME

354                  spyros    Running     1 01:00:00:00  Mon Jul  5 21:21:56
355                  spyros    Running     1 01:00:00:00  Mon Jul  5 21:21:56
356                  spyros    Running     1 01:00:00:00  Mon Jul  5 21:21:56
357                  spyros    Running     1 01:00:00:00  Mon Jul  5 21:21:56

     4 Active Jobs       4 of    4 Processors Active (100.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

358                  spyros       Idle     1 01:00:00:00  Mon Jul  5 21:21:55
359                  spyros       Idle     1 01:00:00:00  Mon Jul  5 21:21:55
360                  spyros       Idle     1 01:00:00:00  Mon Jul  5 21:21:55
361                  spyros       Idle     1 01:00:00:00  Mon Jul  5 21:21:55

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME

Total Jobs: 8   Active Jobs: 4   Idle Jobs: 4   Blocked Jobs: 0
```

Step 5: Install Hashing Implementations

Commands	Additional Details
1. md5sum	(GNU coreutils) 5.97
2. sha1sum	Free Software Foundation
3. sha224sum	/* Written by */
4. sha256sum	- Ulrich Drepper -
5. sha384sum	- Scott Miller -
6. sha512sum	- David Madore -
7. sha1deep	(GPL md5deep) 3.6
8. sha256deep	United States Air Force
9. tigerdeep	/* Written by */
10. whirlpooldeep	- Jesse Kornblum -
11. skein256sum	(GPL PySkein) 0.6
12. skein512sum	/* Written by */
13. skein1024sum	- Hagen Frustenau -

Step 6: Writing the Scripts

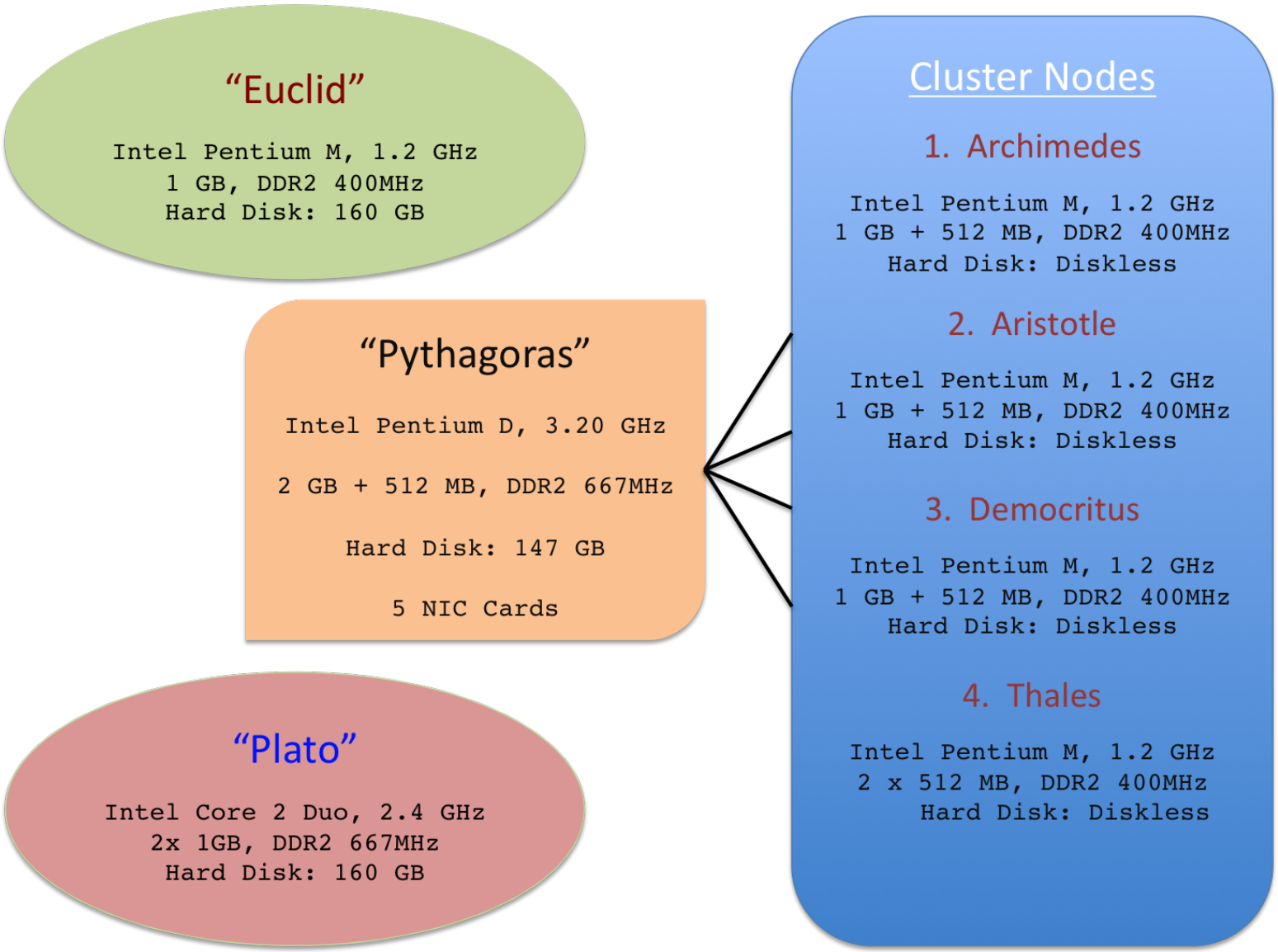
```
#!/bin/bash
# ------------------------------------------------------------------------------
# The following script was used for hashing the entire image database for a single
# algorithm.The sample code corresponds to the md5sum function. Identical scripts
# were used for the other hashing algorithms.
#
# Author: Spyridon Antakis
#
# Notice: The redirection of the output to the "md5sum" file was excluded from the
#         cluster benchmark, due to the existing functionality of the cluster to
#         automatically save the produced output.
# ------------------------------------------------------------------------------

function StartHashing {

cd /home/spyros/database/
date >> ../md5sum
for file in `dir -d *` ;
echo "MD5SUM" >> ../md5sum
/usr/bin/time -f "User:␣%U␣Sys:␣%S␣Time:␣%E␣CPU:␣%P" md5sum "$file" 1>> ../md5sum \
2>> ../md5sum
done
date >> ../md5sum
}

rm -fr /home/spyros/md5sum
StartHashing
```

Cluster & Independent Systems



Compare Performance

Identical Benchmarks in Different Systems

1. Euclid vs Archimedes ⇒ Local vs Remote system
2. Euclid vs Plato ⇒ Slower vs Faster processor
3. Euclid vs Pythagoras ⇒ Slower vs Faster processor
4. Plato vs Cluster ⇒ Faster processor vs Parallelization
5. Pythagoras vs Cluster ⇒ Faster processor vs Parallelization

Test Cases

⇒ Main Goal: Use a variety of workloads with different sizes and analyze the impacts that occur on the diskless architecture.

<table>
<tr><th colspan="2">Different Benchmarks</th></tr>
<tr><th>1. Image Files</th><th>2. Dictionary File</th></tr>
<tr><td>CD Image File (630 MB)</td><td rowspan="3">A list of 10.200 Words

(8.5 bytes per word)</td></tr>
<tr><td>DVD Image File (4 GB)</td></tr>
<tr><td>A Database of Image Files (6.5 GB)</td></tr>
</table>

Formulating 10 early concerns (1)

1. Performance impacts for using a remote & diskless system.
2. Network latency from the *on the fly* processing.
3. Behavior for the different hashing implementations.
4. Performance for the *Skein* algorithm.
5. Processing times & usage of various resources

Formulating 10 early concerns (2)

6. Performance for hashing a dictionary file.

7. Performance of different processors on the same task.

8. Efficiency for splitting a task into the cluster nodes.

9. Advantages & Disadvantages of scaling.

10. Optimizations & additional changes for increasing *performance*.

Hashing a CD Image

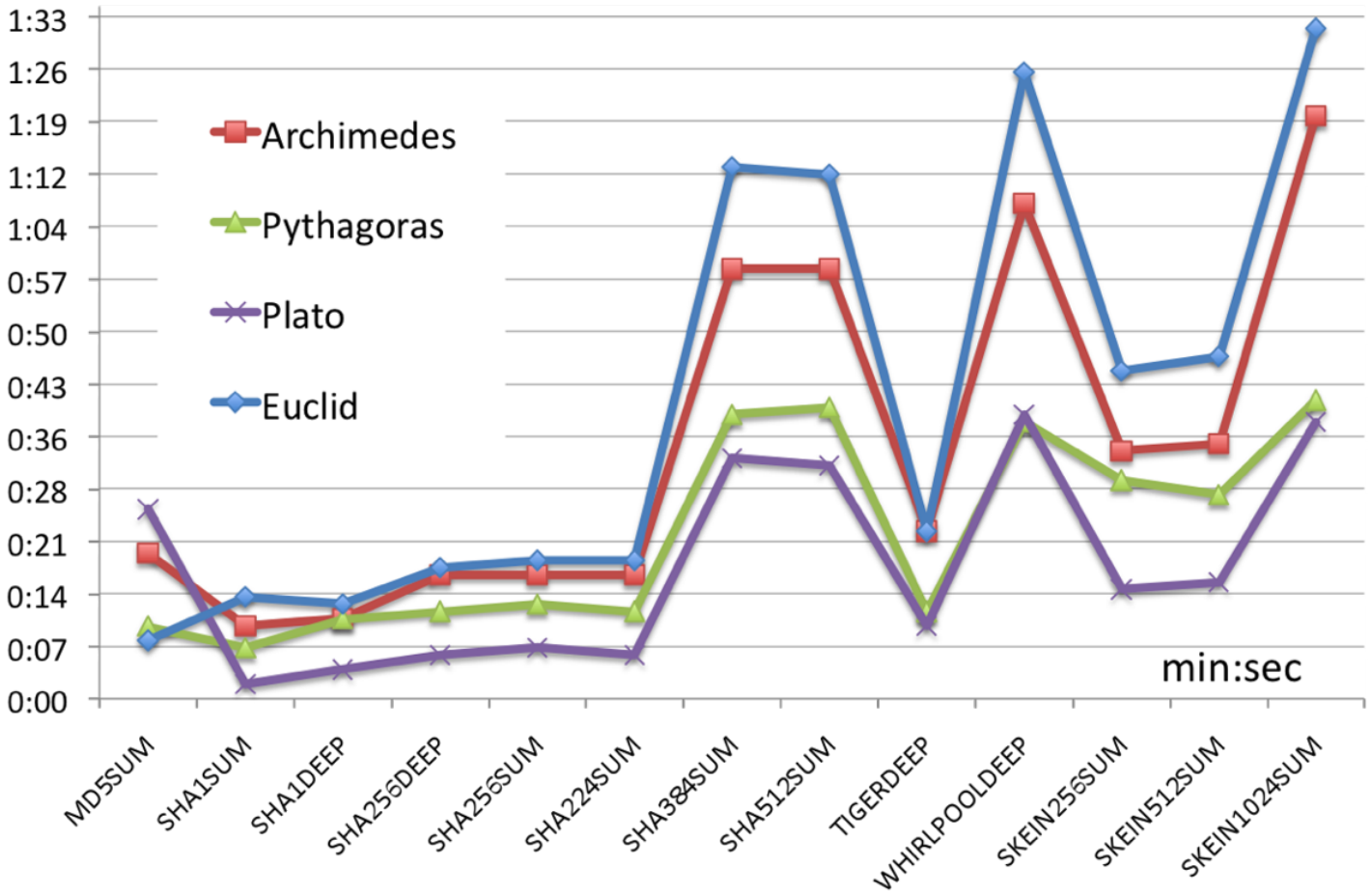


Figure: Hashing a CD Image (630 MB)

Hashing a DVD Image

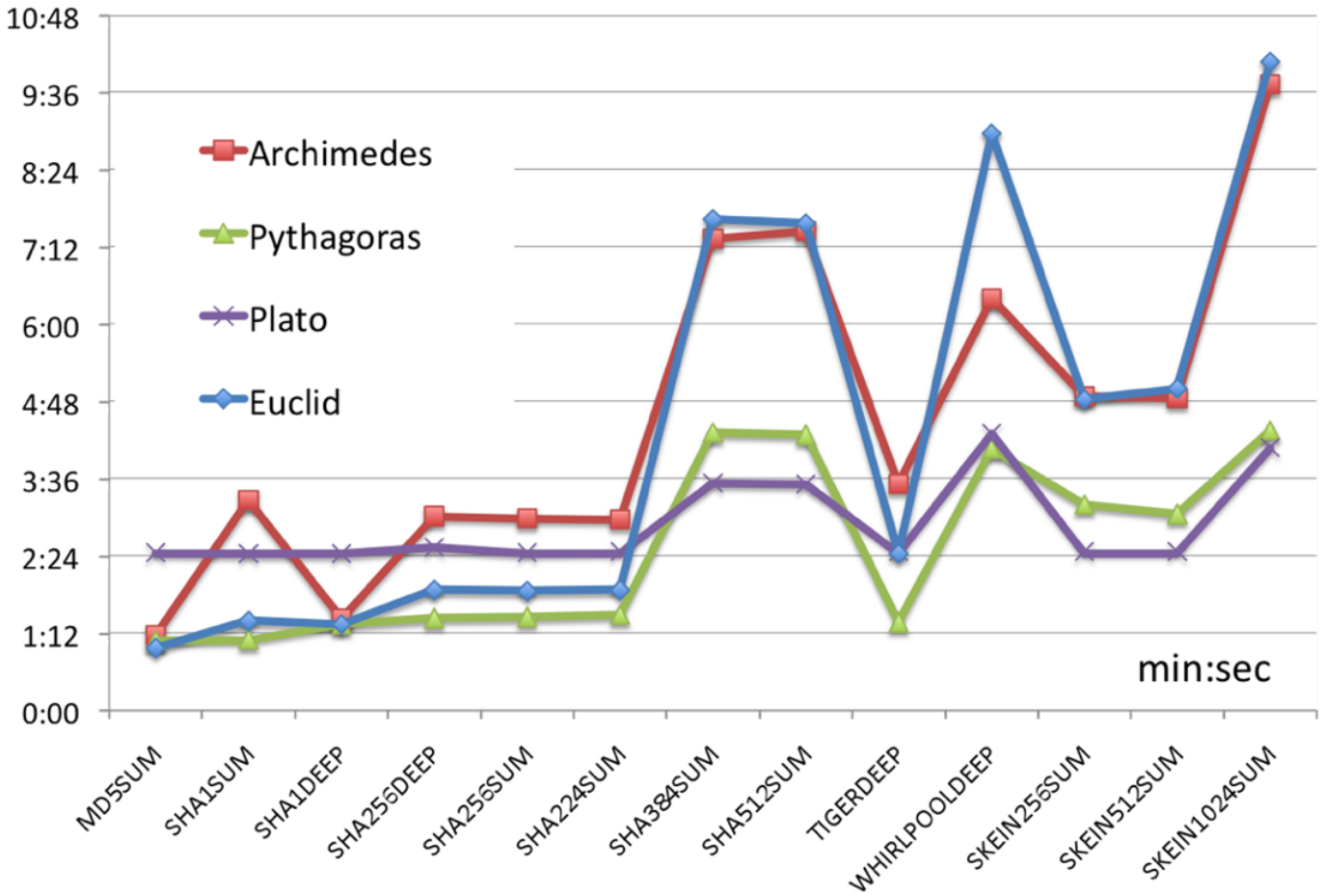


Figure: Hashing a DVD Image (4 GB)

Hashing a Database of Image Files

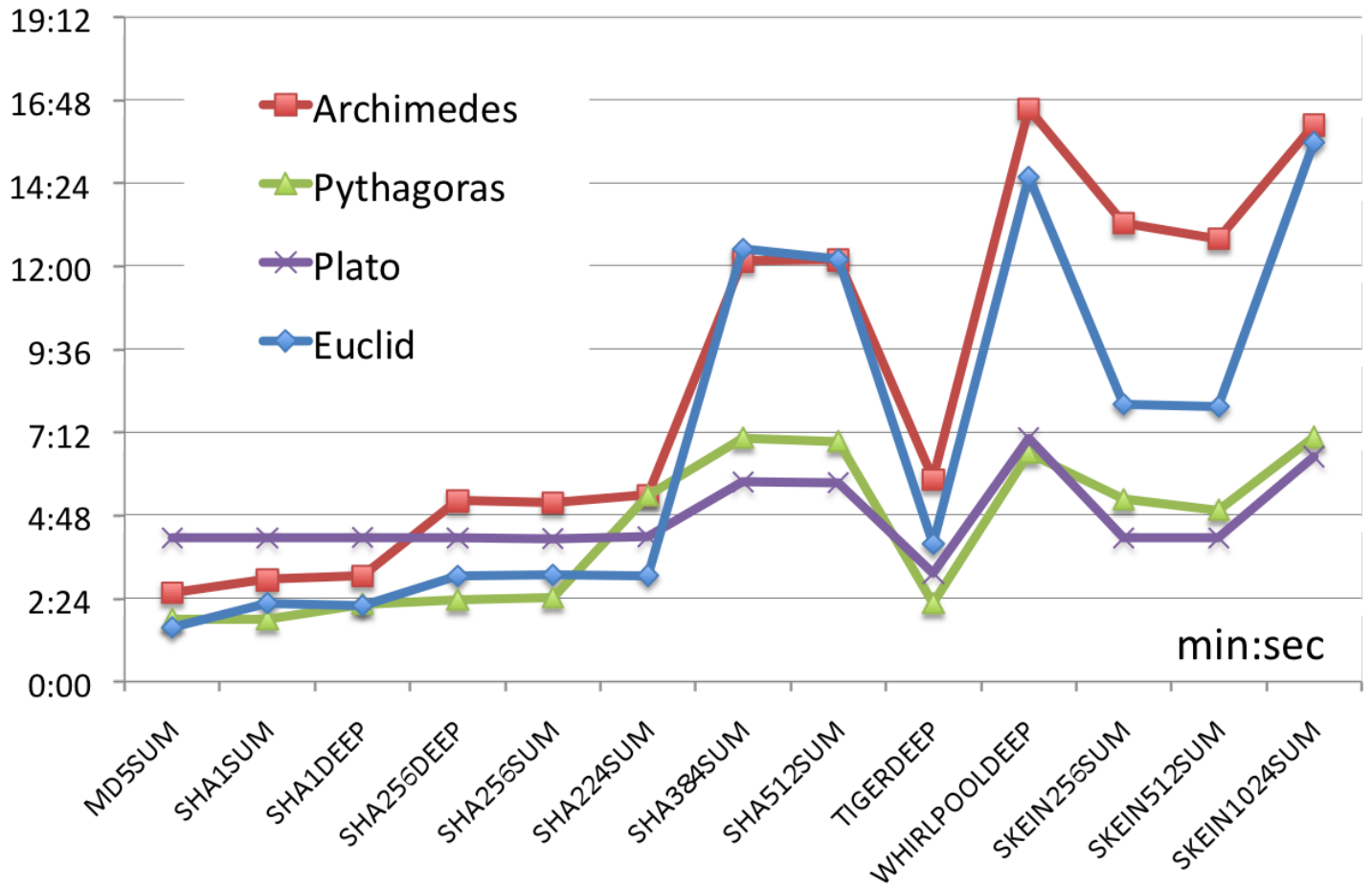


Figure: Hashing the Database (6,5 GB)

Cluster vs Units (Image Files)

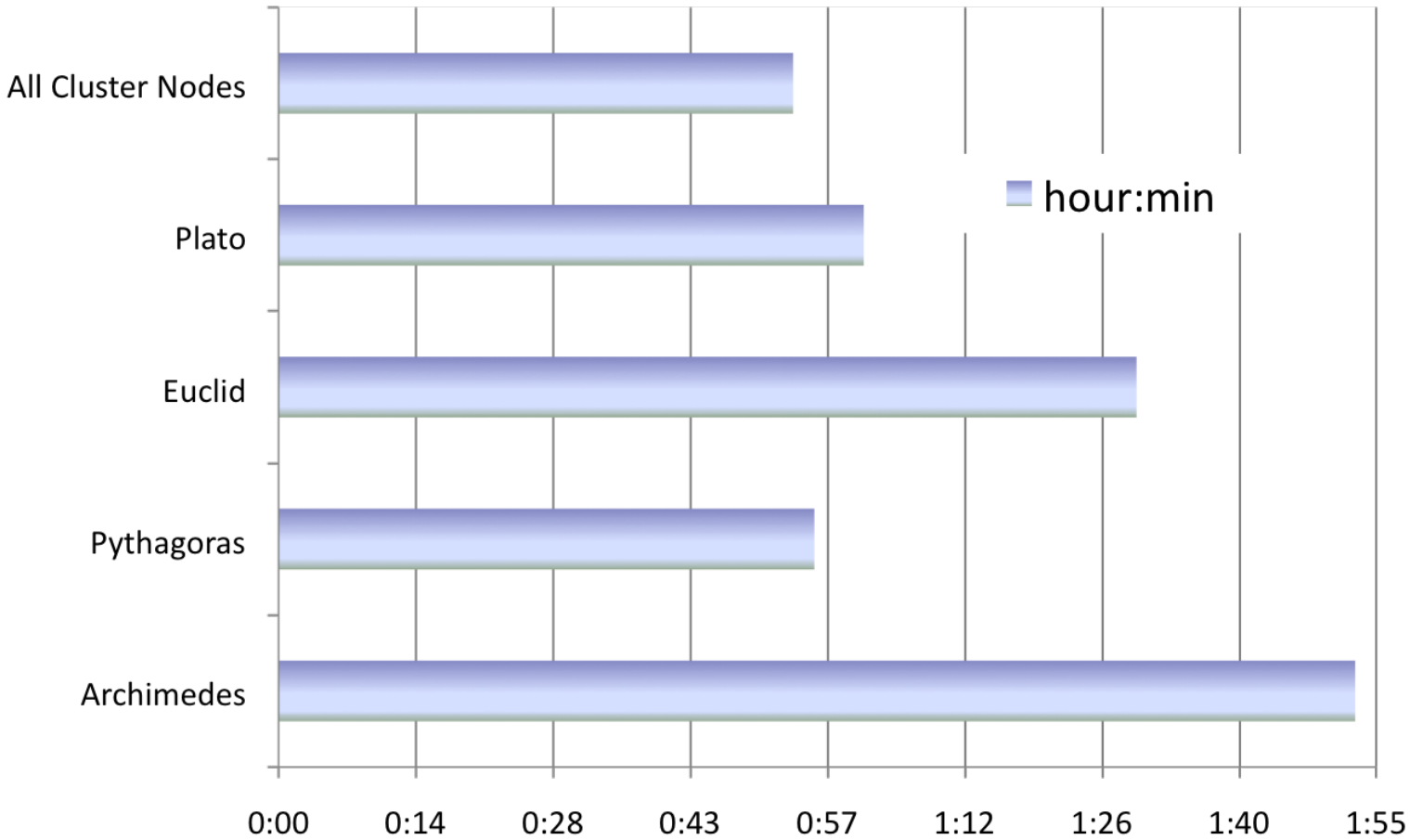


Figure: Cluster vs Units (entire database - 6,5 GB)

Hashing a List of Words (1)

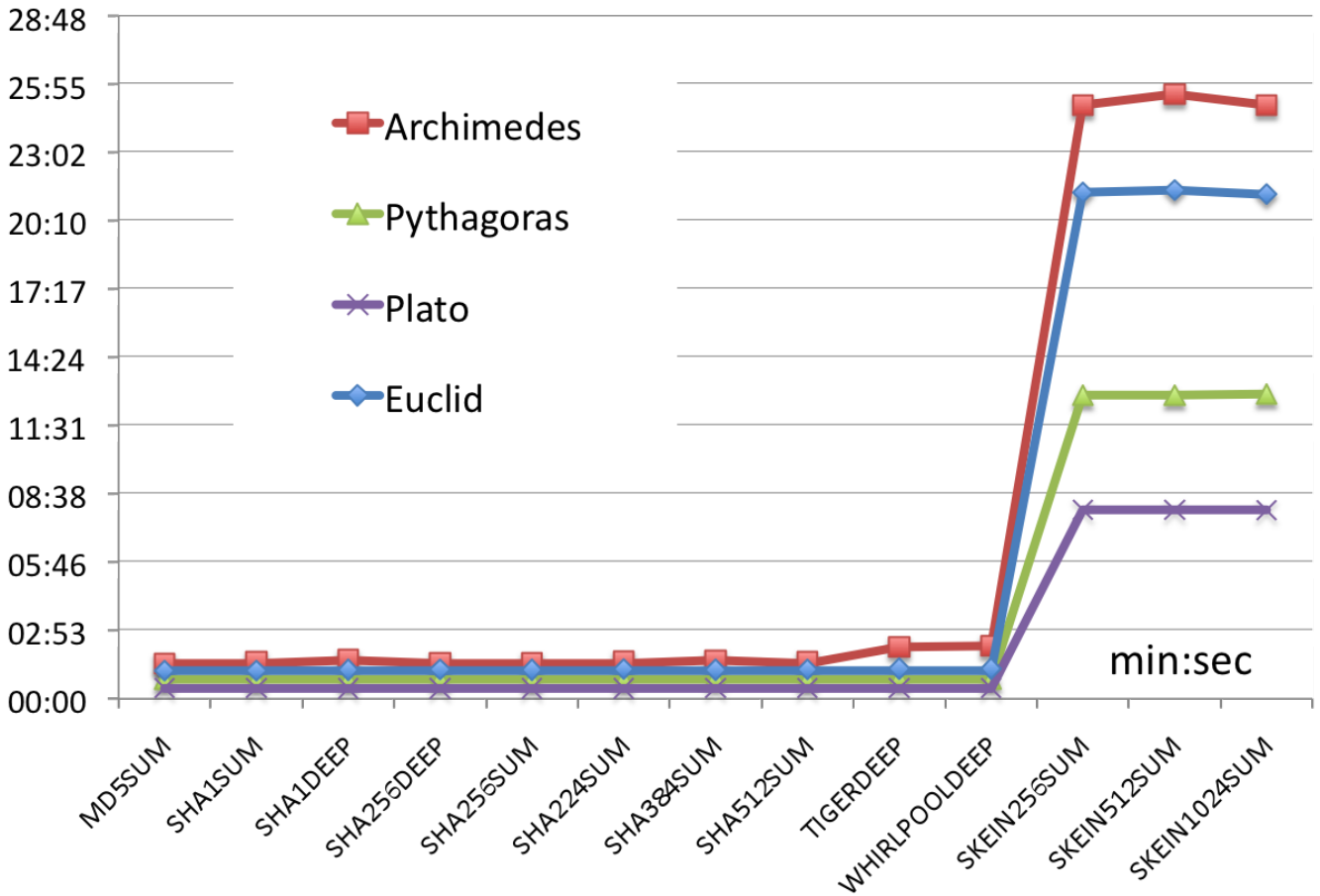


Figure: 10.200 Words

Hashing a List of Words (2)

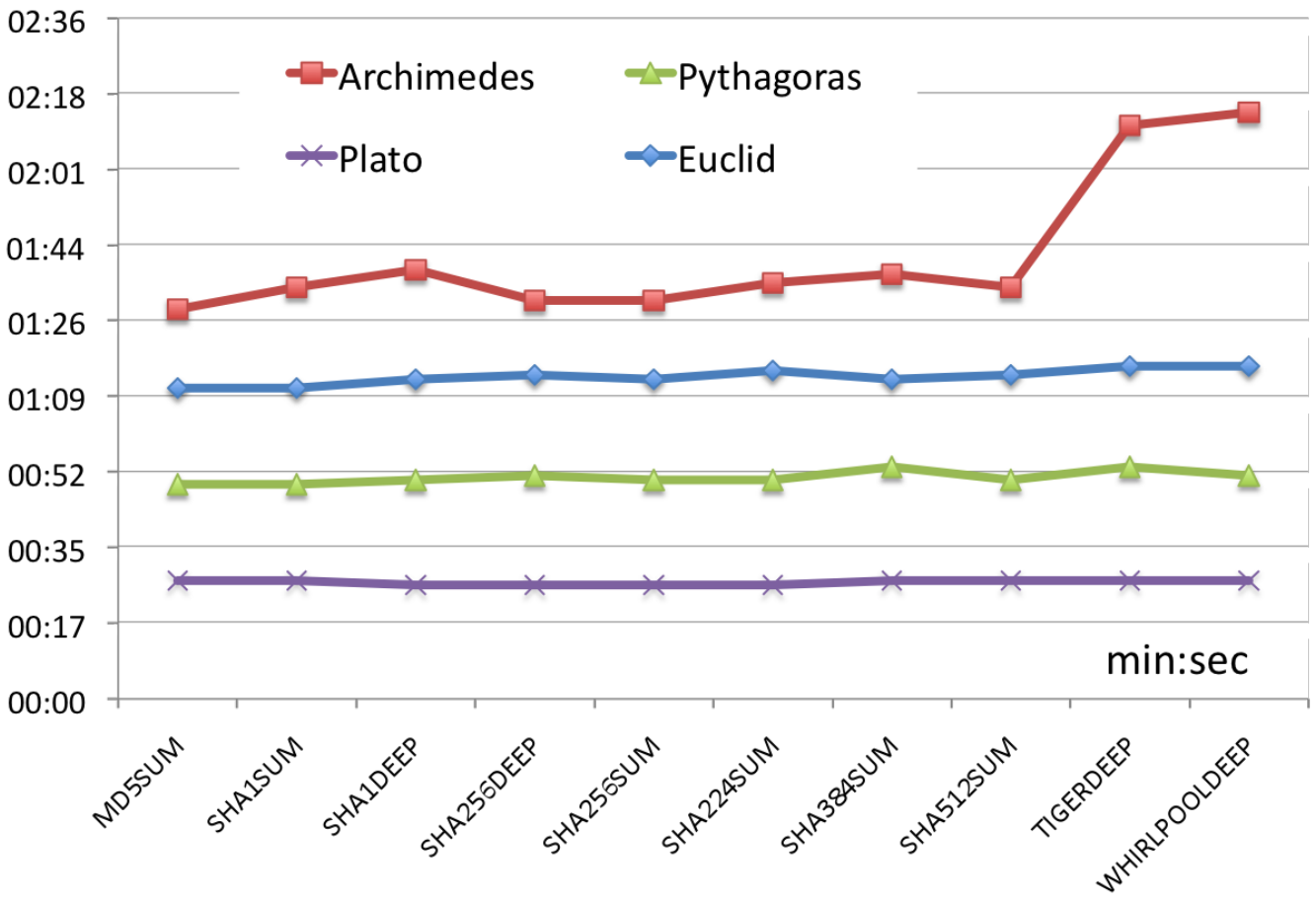


Figure: 10.200 Words - Zoomed Graph

Cluster vs Units (Dictionary File)

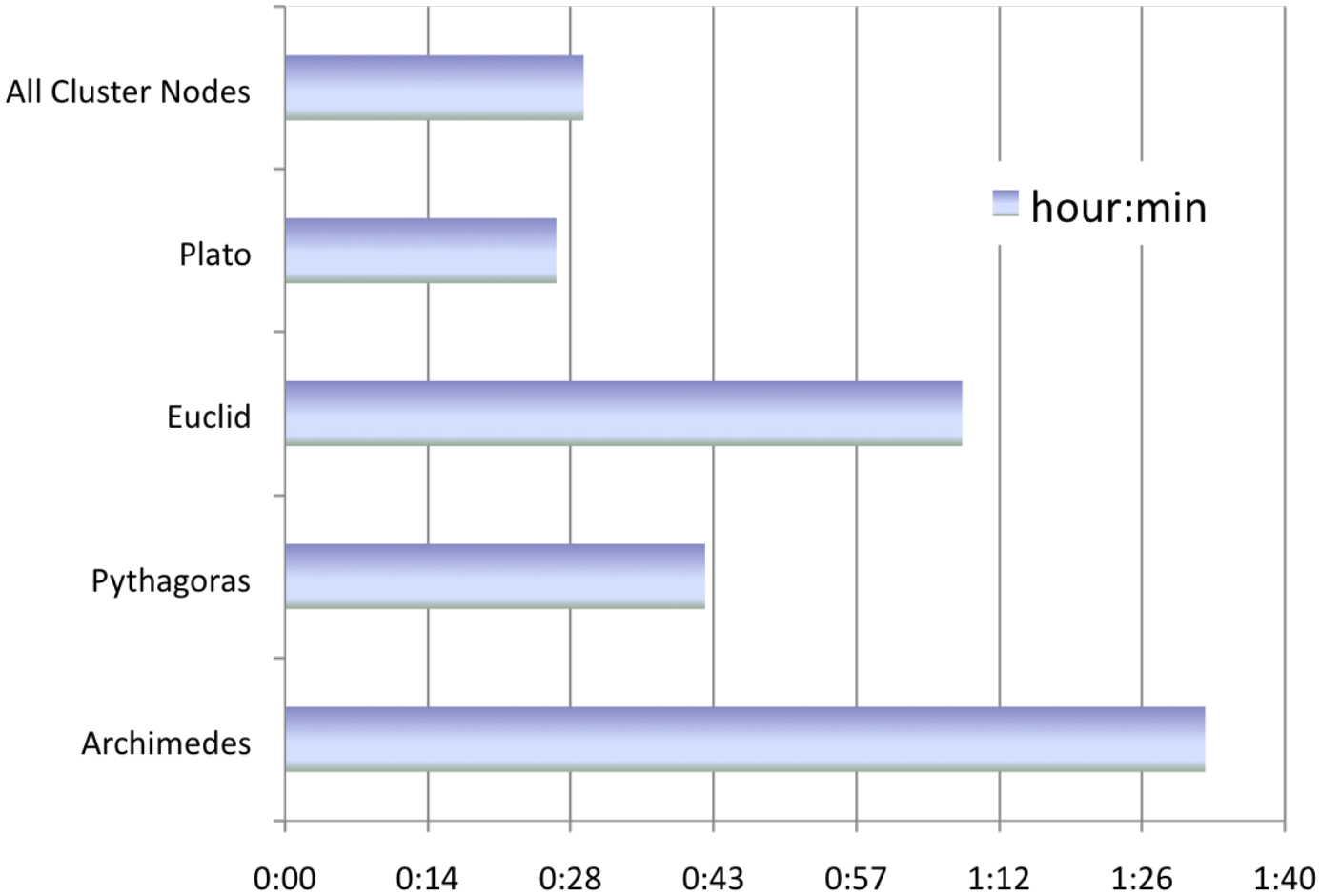


Figure: Cluster vs Units (10.200 words)

Conclusions

How promising is this architecture ?

- Parallelization to the cluster nodes could be improved.
- Acceleration features could boost the processing power.
- An alternative solution for dedicated purposes.
- Additional investigation for this model is necessary.
- The value of the architecture lies on the nature of the tasks.

Conclusions

Which are the major challenges ?

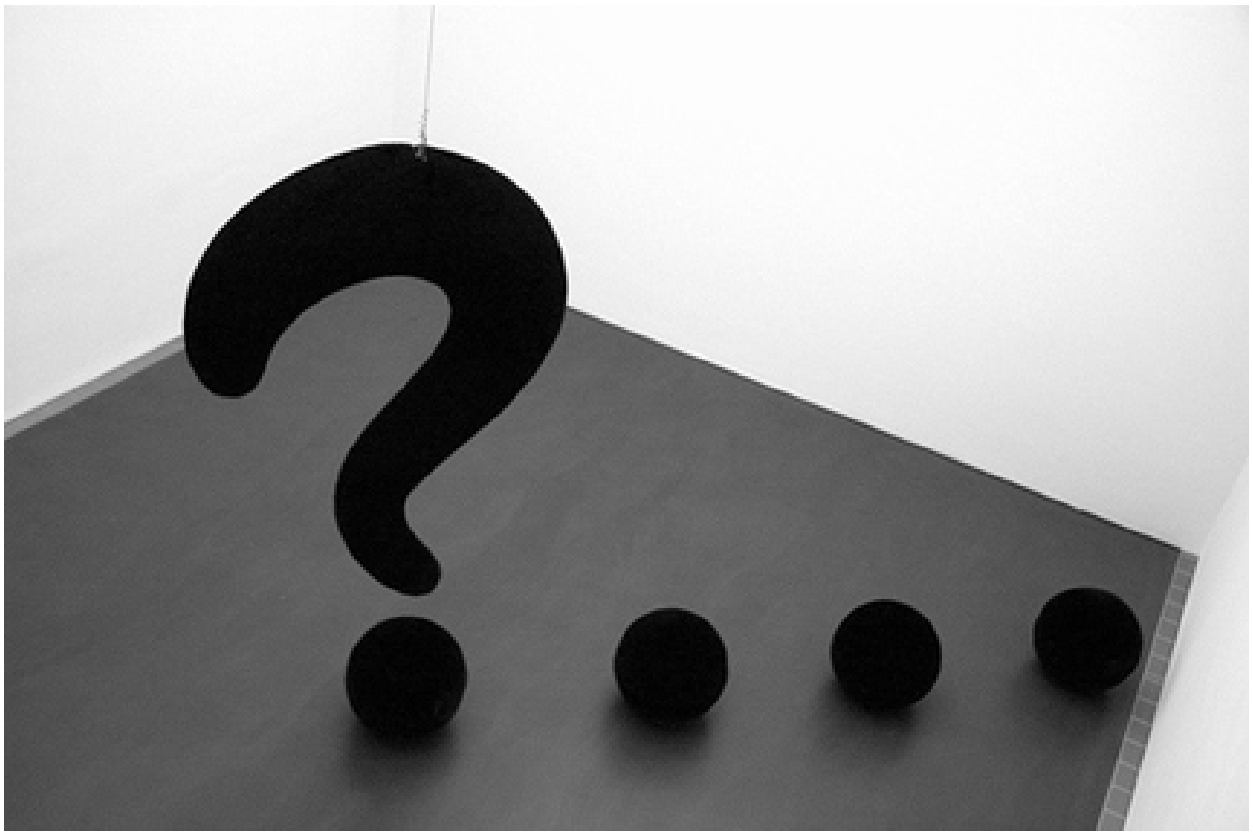
- Control the CPU utilization.
- Parallelize with the most optimal way (e.g. OpenMPI).
- Minimize the remote access calls to the shared hard disk.
- Investigate the limitations of the data transfer protocols.
- Use low level languages, such as C/Assembly.
- Mitigate the trade-offs between cost & performance.

Conclusions

What about advantages ?

- The model is quite flexible.
- The architecture can be deployed on normal network infrastructures.
- There are several open source tools for clustering.
- Parallelization becomes straightforward.
- GPUs or Acceleration, could constitute additional features.
- A mixed *hardware* and *software* environment is possible.

Questions



The dissertation is available here: http://www.s-antakis.gr