Hadoop

Jörg Möllenkamp
Principal Field Technologist

Sun Microsystems

Agenda

Introduction
CMT+Hadoop
Solaris+Hadoop
Sun Grid Engine+Hadoop

Introduction

I‘m ...

Jörg Möllenkamp

better known as „c0t0d0s0.org“

Sun Employee

Principal Field Technologist

from Hamburg

I'm ...

Jörg Möllenkamp

better known as „c0t0d0s0.org"

Sun Employee

Principal Field Technologist

Hamburger (Open)Solaris User Group

thus a part of the HHOSUG as well ...

An apologize right at the start …

No live demonstration ...

....Sorry

Had a „shortnotice“ customer meeting at 10:00 o‘clock ...

3 presos yesterday, one this morning.
so my voice may be a single point of failure ...

Or to say it with Rudi Carrell
„A moment ago in a meeting room in Bremen, now on the stage in Berlin“

Had no time to test my „demo case“ ....

And i‘ve learned a thing in thousand presos:
Never ever do a live demo without tests ...
... will ruin your day big time ...

In the scope of this presentation:

Why is Sun interested in Hadoop?

Mutual significance

A little bit bragging about some new Sun HW

Not in the scope of this presentation:

Explaining you the idea behind Hadoop

The History of Hadoop

Just providing a list of Sun Hardware


SunFire X4540
Sun
microsystems


CAUTION
CAUTION
DO NOT LIFT
DO NOT LIFT


Sun Fire X4540
System Controller (SC)
CAUTION
CAUTION

Sun+Hadoop

Why is Sun working with Hadoop?

At first: It‘s an „I“ technology.

Not „I“ for „Internet“

„I“ for „Interesting stuff“

At the CEC2008

Hadoop was an important

part on the Global Systems Engineering Tracl

We think that:
Hadoop can provide something to Sun
But as well:
Sun can provide something to Hadoop

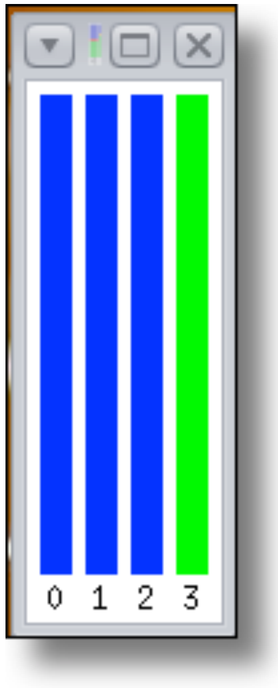
Hadoop+CMT

What can Hadoop provide for Sun?

Another usecase for a special kind of hardware

CMT
Chip Multi Threading

4 or 8 Cores are for Sissys



2005

UltraSPARC T1

8 Cores

4 Threads per Core

32 Threads per System

2007

UltraSPARC T2

8 Cores

2 Integer Pipelines per Core

4 Threads per Pipeline

64 Threads per CPU

2008

UltraSPARC T2+

CMT goes SMP

8 Cores

2 integer pipelines per core

4 threads per pipeline

64 Threads per CPU

4 CPUs per system

256 threads per system

2010

UltraSPARC „Rainbow Falls“

16 Cores

2 integer pipelines per core

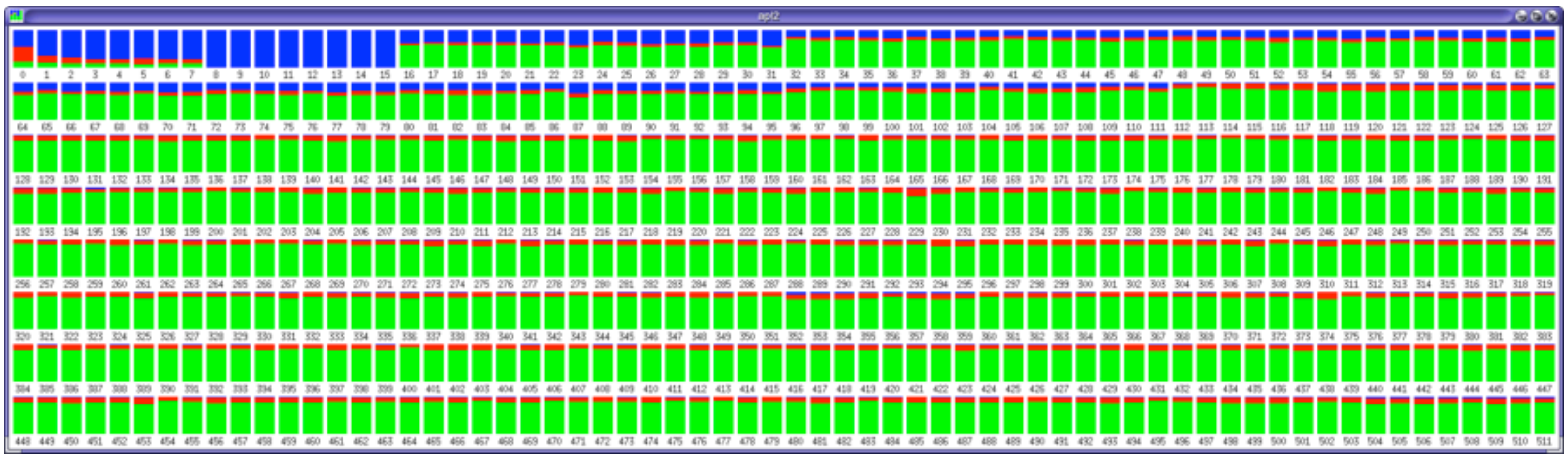
4 threads per pipeline

128 Threads per CPU

4 CPUs per system

512 threads per system

That would look like that:



obviously a single grep process
don’t scale that well on this system ...

Those system eat threats ... lot‘s of them ...

Otherwise it's relatively slow ...
just 1.6 GHz at the moment.

But 4 memory controllers today, more later ...
because frequency means nothing if your proc has to wait for data from RAM ...

Or perhaps a better analogy ...
It doesn‘t matter if you stir your diner at
1.6 GHz or 4.7 GHz
when you have to wait for
your significant other
to get the bottle of wine from the cellar.

To be honest ...

my colleagues made the last screenshot on this system

We have an operating system
that can use this amount
of threads.

But that‘s only half of the story:
You need applications that are able to generate the load.

UltraSPARC Tx is a massively parallel, throughput centric architecture ...

Sound familiar?

Yes … indeed!

Would you like your Hadoop in a box?

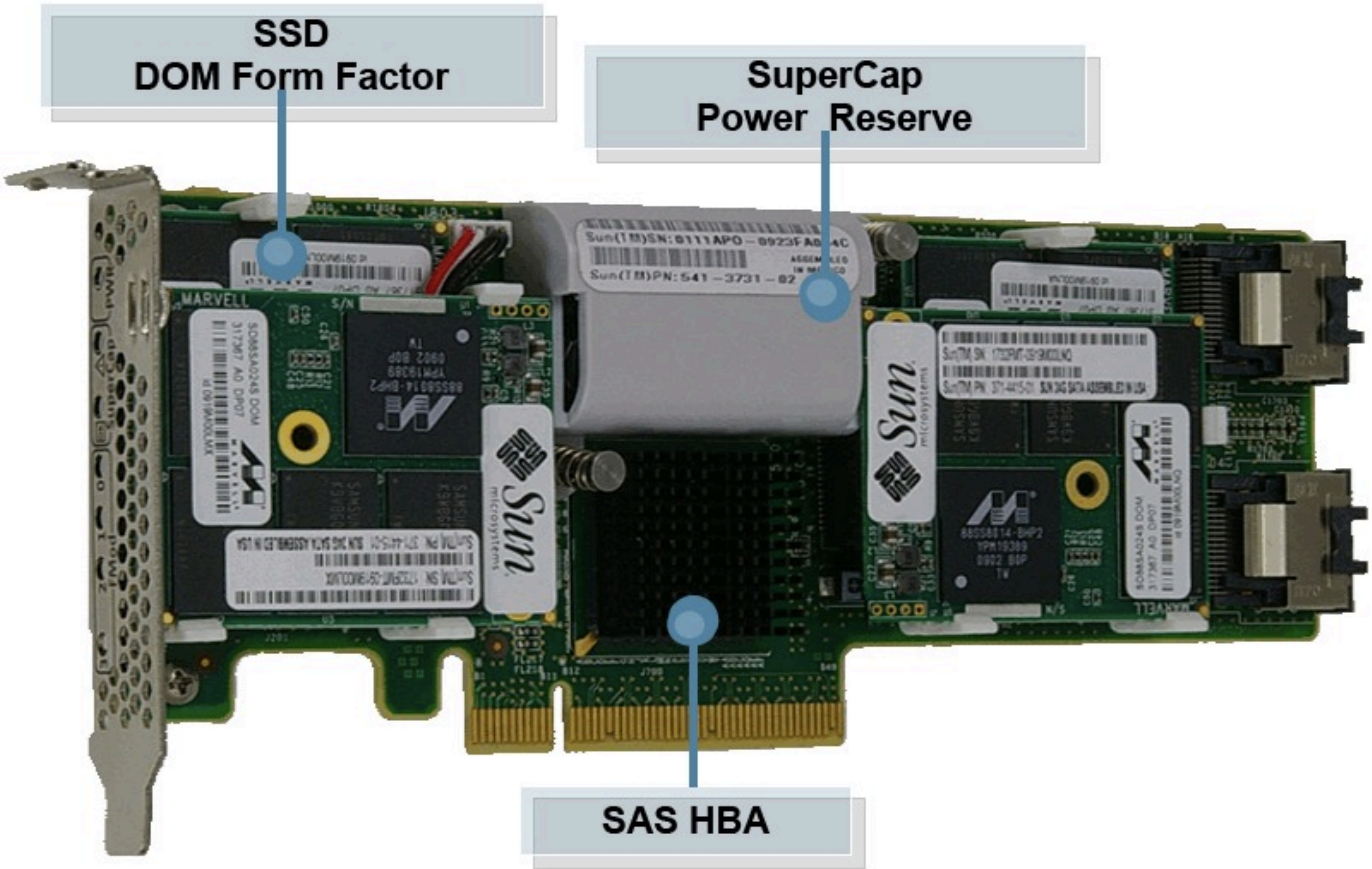
Wasn't Hadoop developed with small boxes in mind?

Yes ... of course.
But there is still a reason for using T-Class systems.

Density!

	Yahoo* 40*1U	Blade 6000 with T2 blade	T5240	T5440+J4400
Size	40*1U	4*10U	20*2U	5+5x4U
Thread/Node	8	64	128	256
Disks/Node	4	4	16	24
Memory/Node	8-16 GB	128 GB	256 GB	512 GB
Nodes/Rack	40	40	20*2U	5
Threads/Rack	320	2560	2560	1280
Memory/Rack	320-640 GB	5120 GB	5120 GB	2560 GB
Disks/Rack	160	160	2320	120

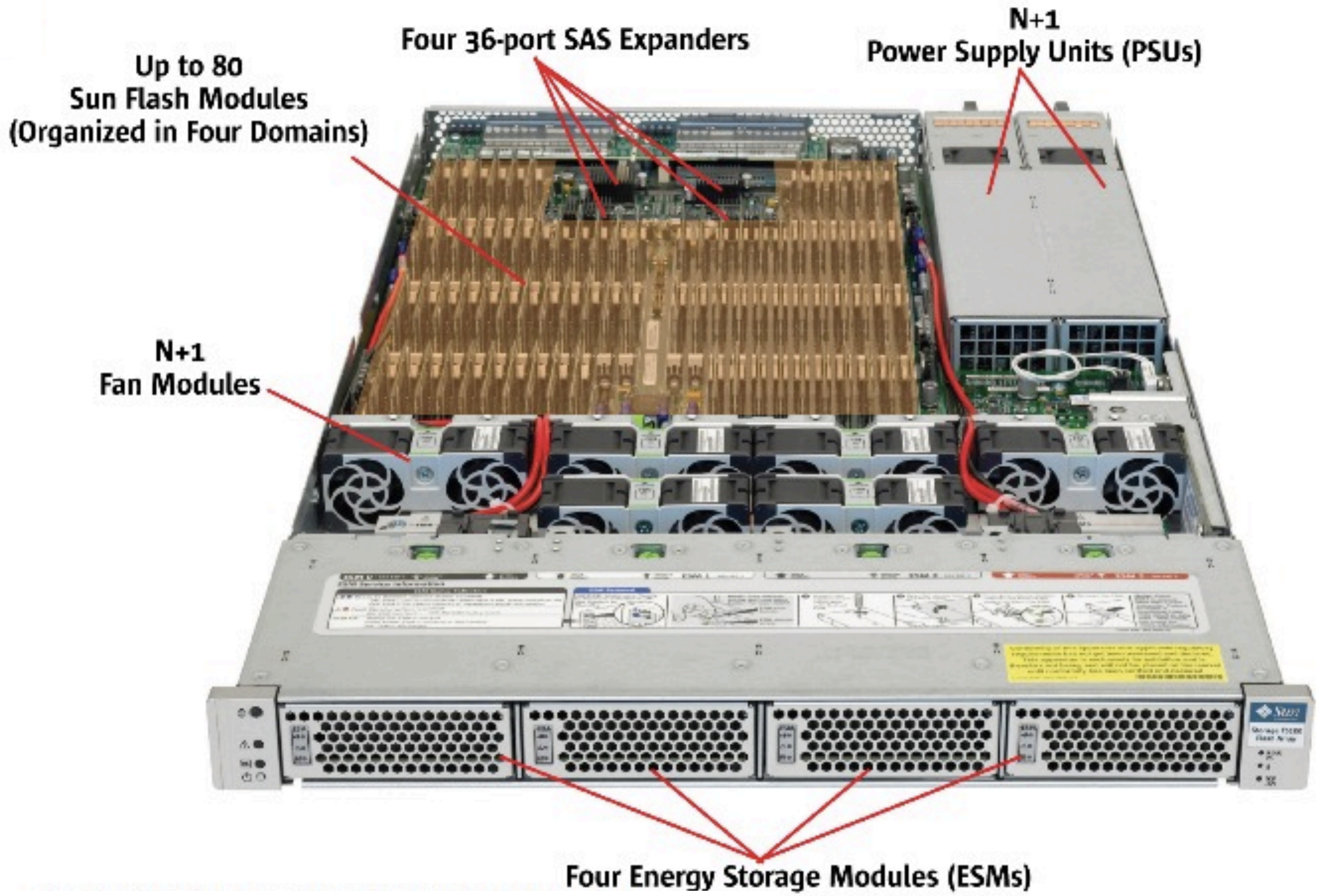
More density? More performance?


SSD
DOM Form Factor
SuperCap
Power Reserve
SAS HBA

Capacity		
	48 GB	96 GB
Domains	2	4
Flash Type (NAND)	Single Level Cell (SLC)	Single Level Cell (SLC)
Performance[1]		
	48 GB	96 GB
Random Read IOPS (4 K)	53 K IOPS	100 K IOPS
Random Write IOPS (4 K)	42 K IOPS	87 K IOPS
Sequential Read Rate (1024 K transfer size)	546 MB/s	1092 MB/s
Sequential Write Rate (1024 K transfer size)	250 MB/s	494 MB/s

When you want to go really extreme ...

Sun Storage Flash Array F5100


Up to 80
Sun Flash Modules
(Organized in Four Domains)
Four 36-port SAS Expanders
N+1
Power Supply Units (PSUs)
N+1
Fan Modules
Four Energy Storage Modules (ESMs)

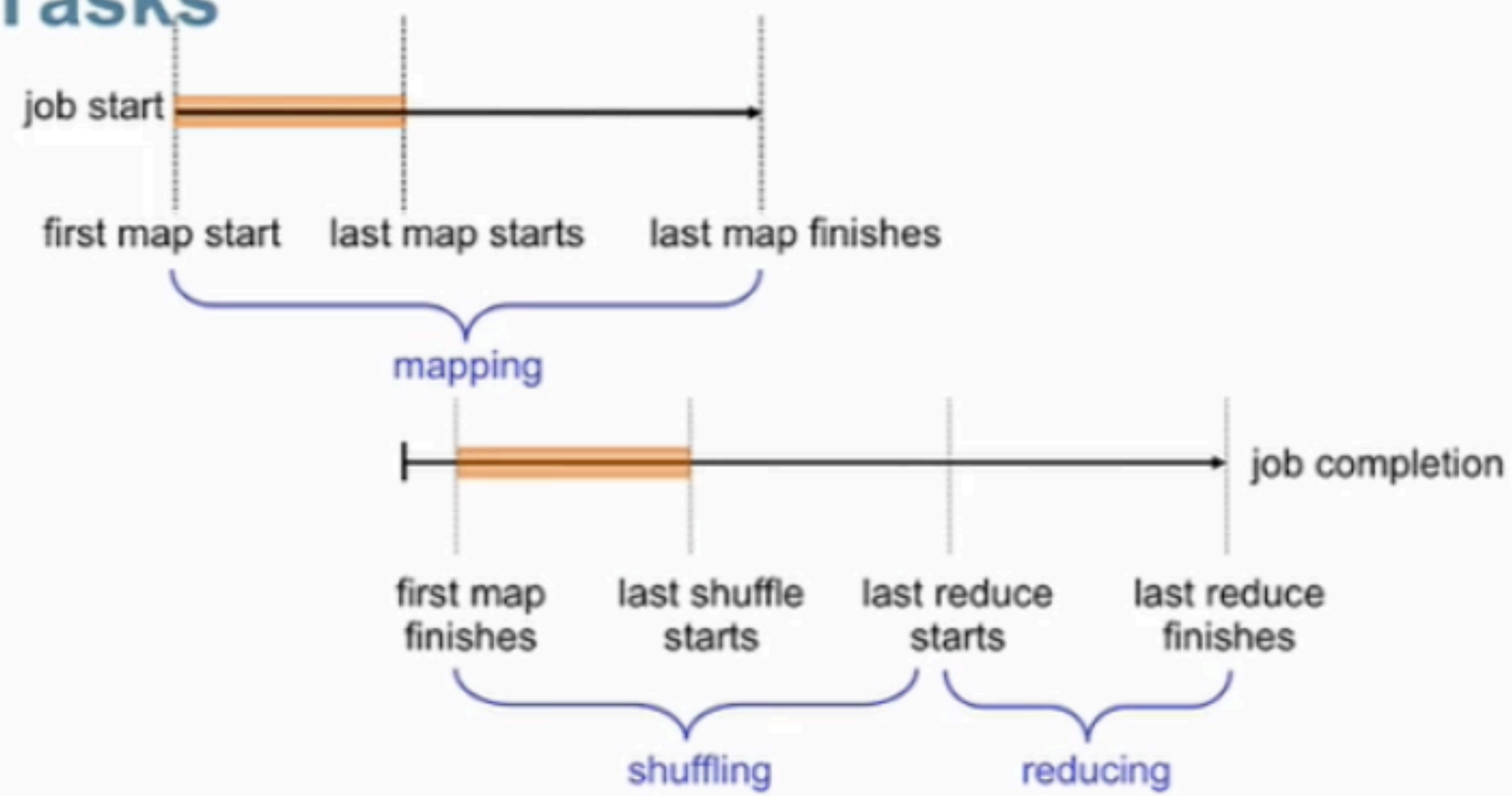
1 rack unit
1.2 million IOPS random write
1.6 million IOPS random read
12.8 GByte/s sequential read
9.6 GByte/s sequential write
1.92 TB capacity

	Yahoo* 40*1U	Blade 6000	T5240	T5440+J4400	T5440+F5100	T5120+F5100
Size	40*1U	4*10U	20*2U	5+5x4U	8*(1U + 4U))	20*(1U+1U)
Thread/ Node	8	64	128	256	256	128
Disks/ Node	4	4	16	24	80	80
Memory/ Node	8-16 GB	128 GB	256 GB	512 GB	512	256
Nodes/ Rack	40	40	20	5	8	20
Threads/ Rack	320	2560	2560	1280	2.048	2560
Memory/ Rack	320-640	5120 GB	5120 GB	2560 GB	4.096	5120
Disks/ Rack	160	160	320	120	640	1600

But colleagues found
a problem with such large cluster

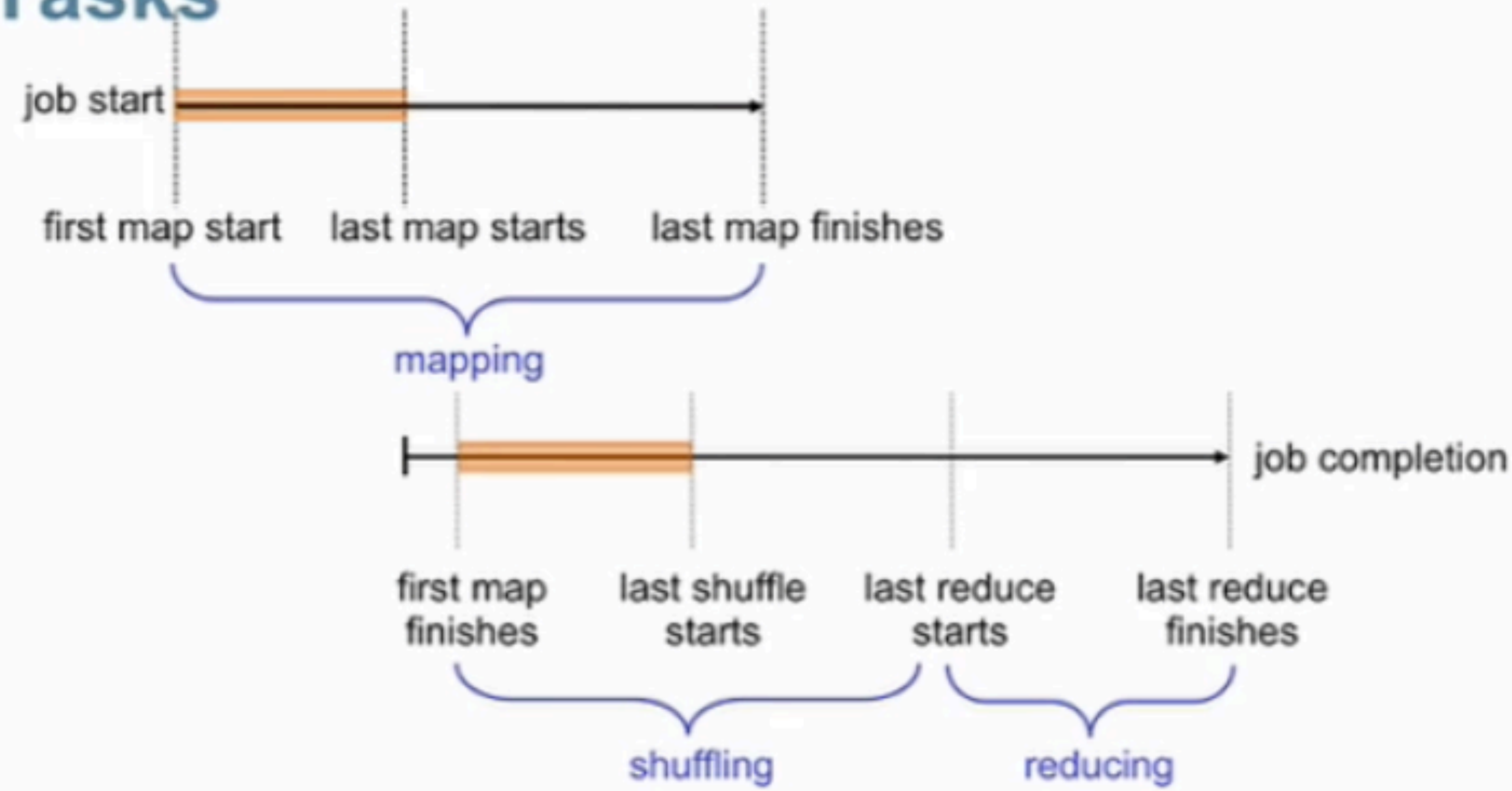
I will just use their slides now ...

Performance Model with Serialized Tasks



Launching many tasks can incur significant overhead

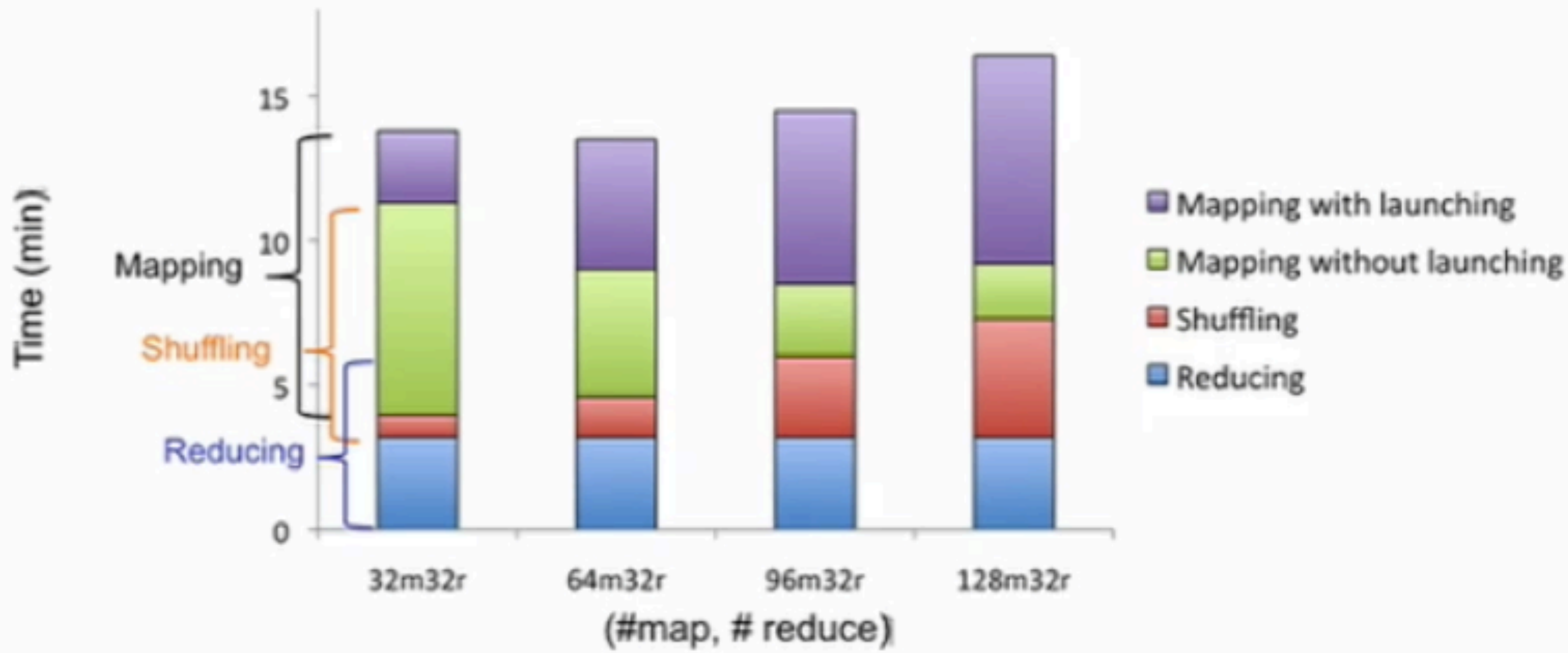
Performance Model with Serialized Tasks



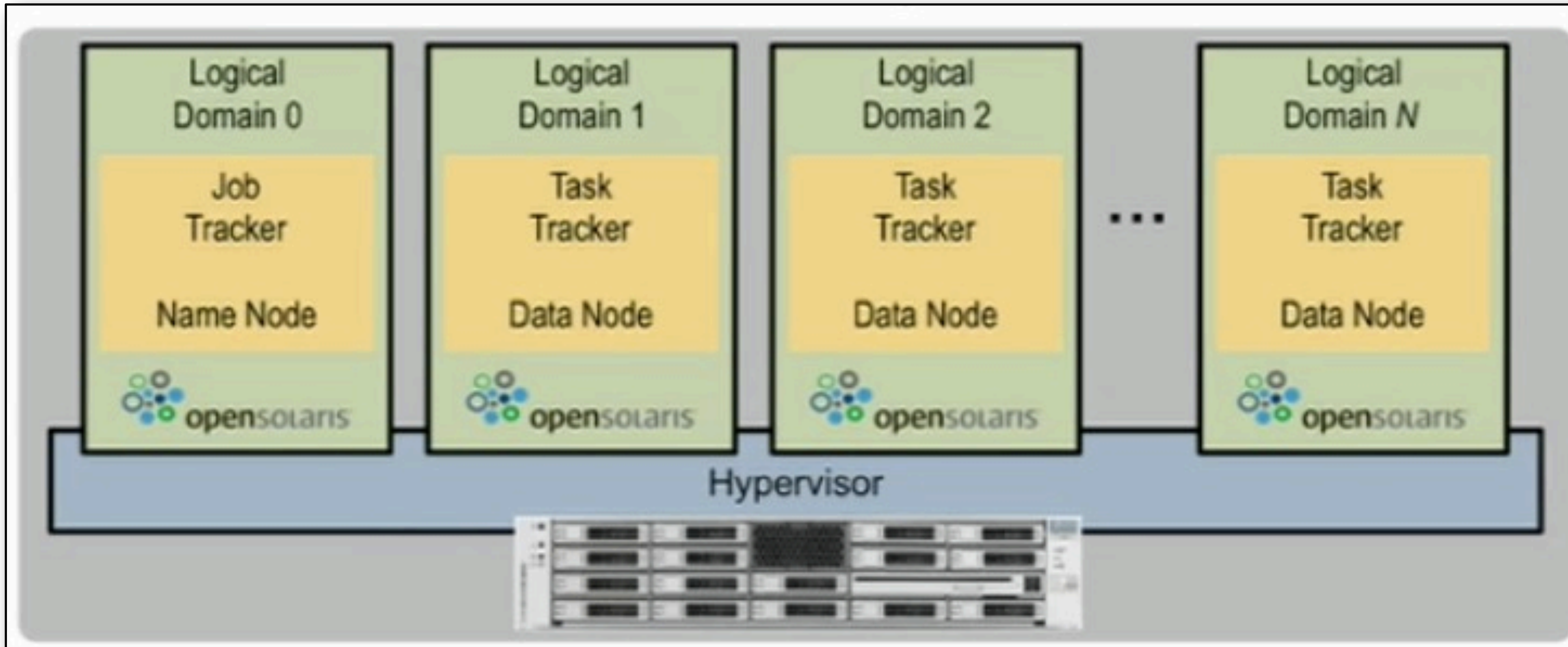
Launching many tasks can incur significant overhead

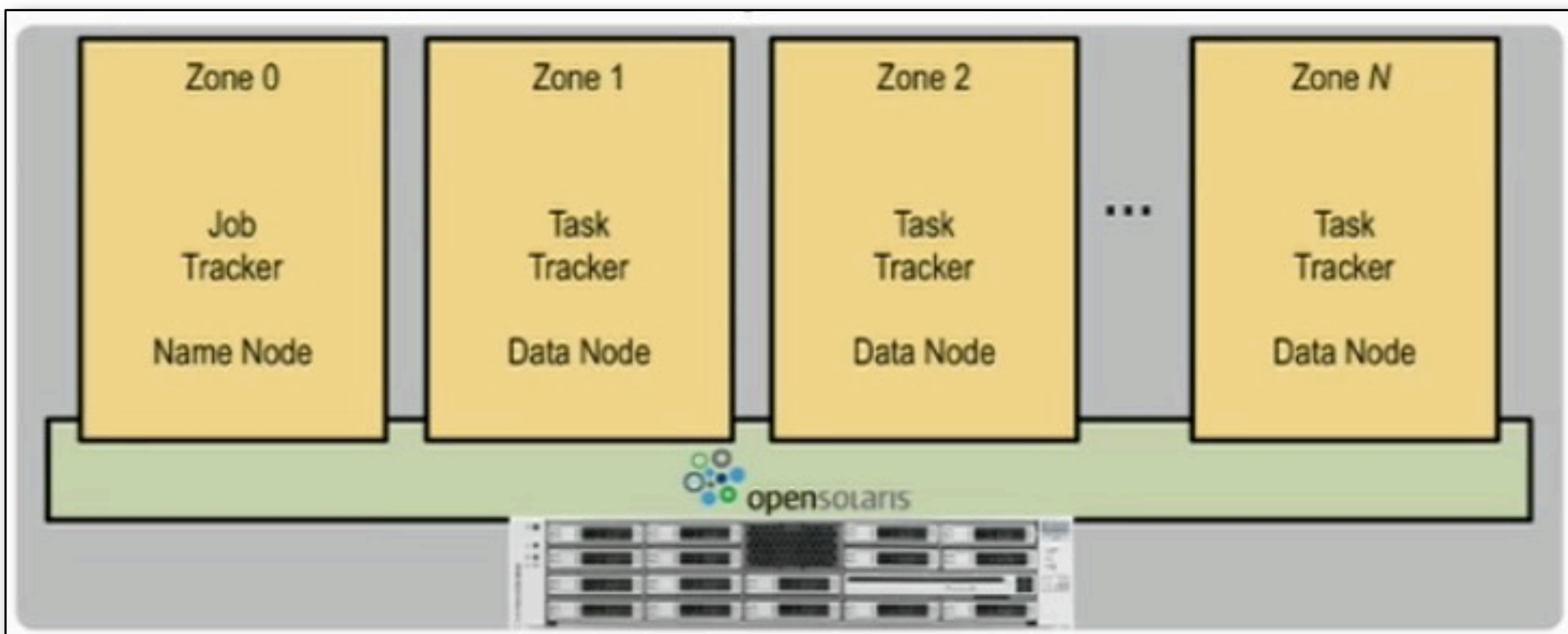
Serialized Task Launching Overhead

30GB sort on a single T5240 node (128 threads, 128GB RAM, 16 disks)

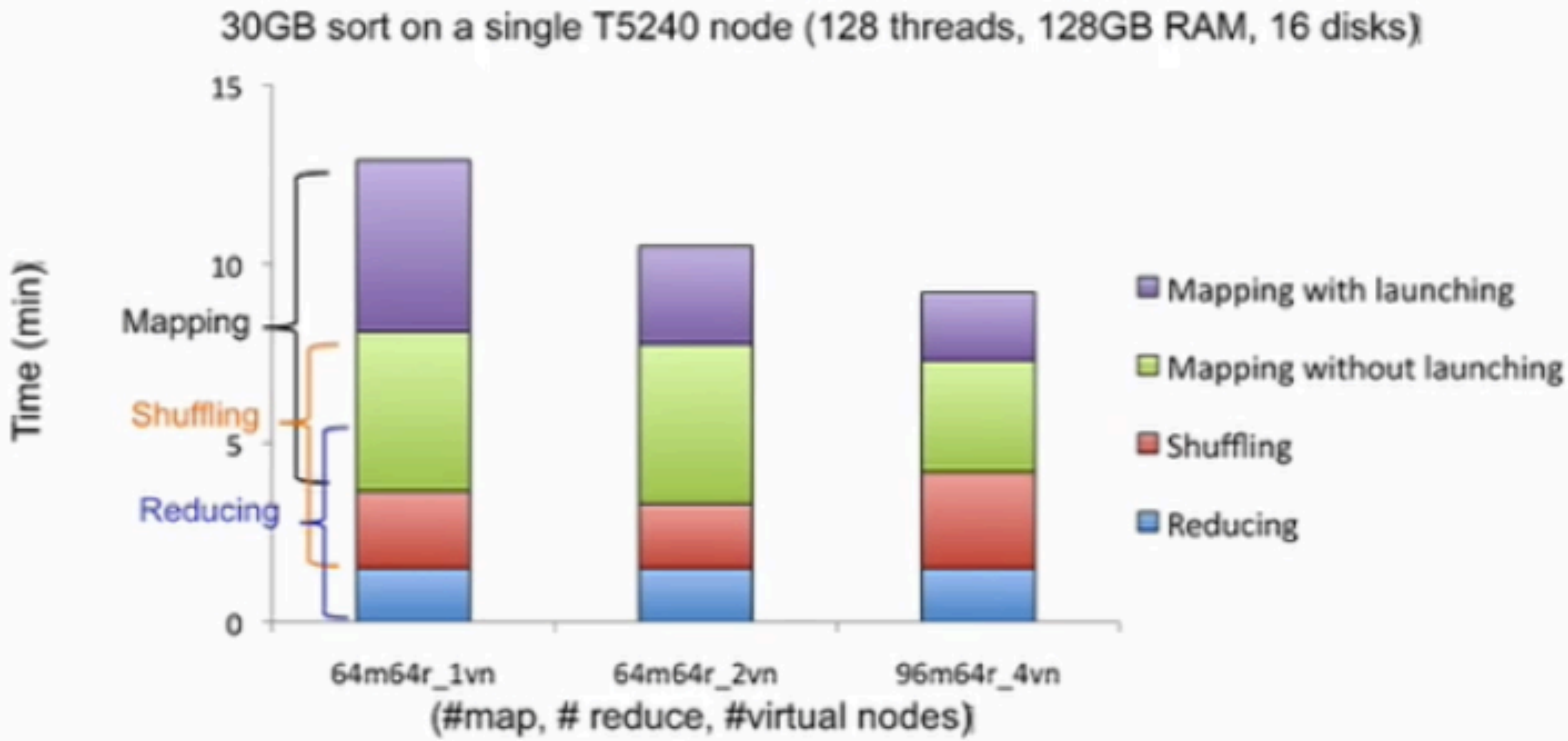


<60% CPU utilization
Significant launching overhead limits scalability


Logical Domain 0
Job Tracker
Name Node
opensolaris
Logical Domain 1
Task Tracker
Data Node
opensolaris
Logical Domain 2
Task Tracker
Data Node
opensolaris
...
Logical Domain N
Task Tracker
Data Node
opensolaris
Hypervisor

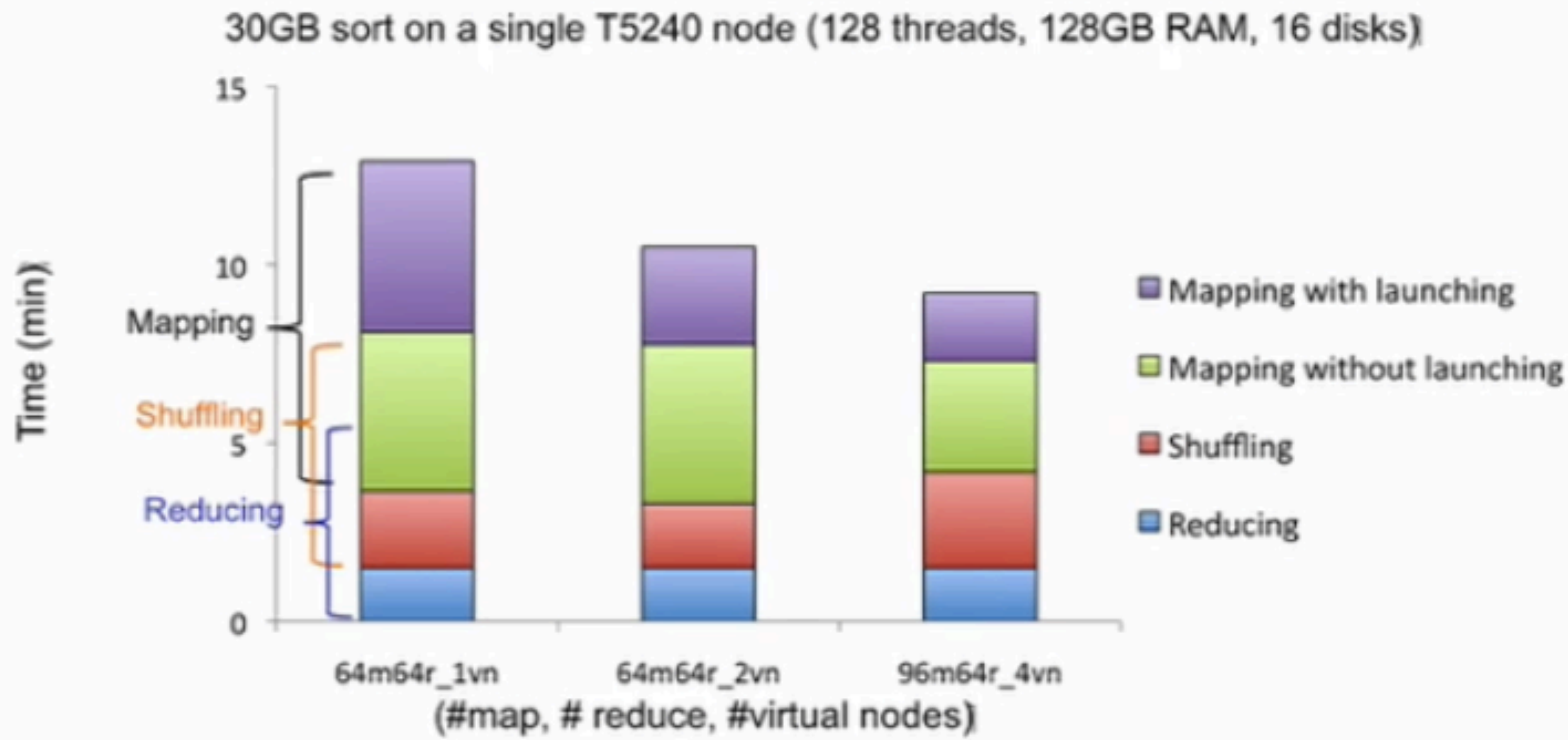

Zone 0
Job Tracker
Name Node
Zone 1
Task Tracker
Data Node
Zone 2
Task Tracker
Data Node
...
Zone N
Task Tracker
Data Node
opensolaris

Scaling Hadoop with Intra-node Virtualization



~100% CPU utilization with 4 logical domains

Scaling Hadoop with Intra-node Virtualization

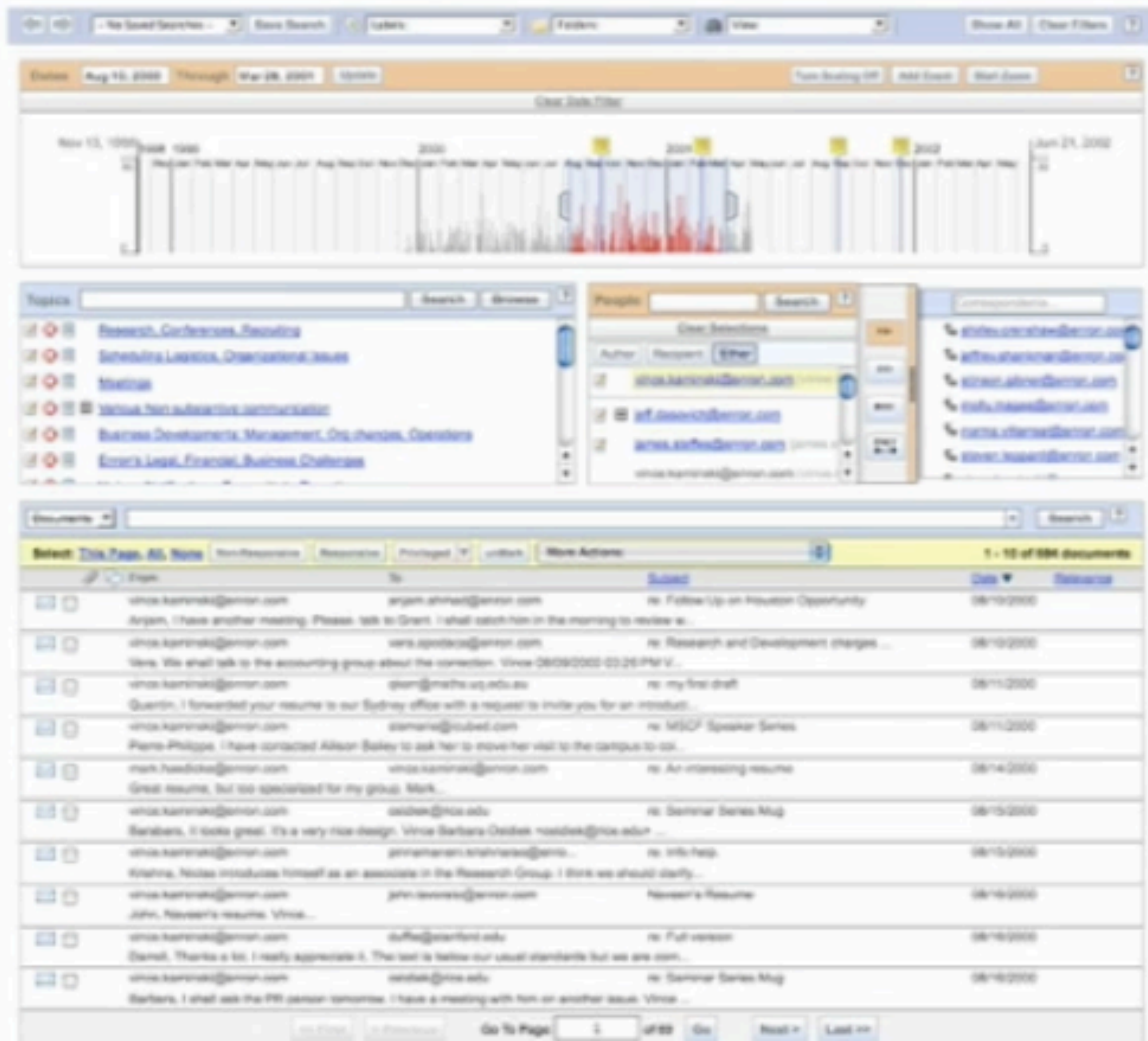


~100% CPU utilization with 4 logical domains

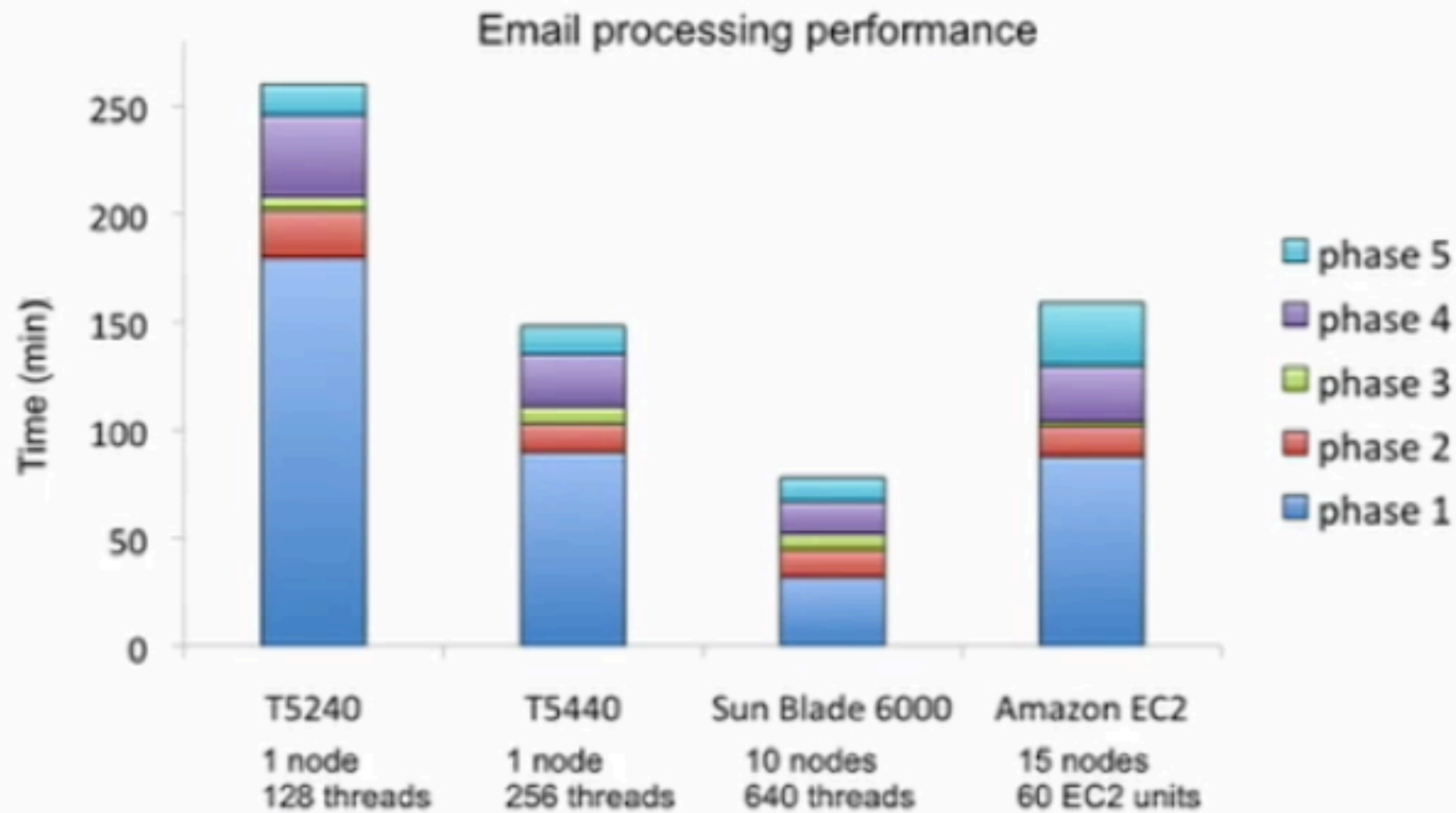
E-mail Discovery Overview

- Preparing data for searching over large email corpus
- Five phases with different MapReduce profiles
  1. PipelineMapReduce – Reads and parses 27GB of raw emails
  2. DocumentSeqFileToMapFile – Prepares MapFile to retrieve data
  3. PersonNormalization – Groups data into unique entities
  4. Consumer – Creates indices
  5. ThreadDetection – Conversation threads detected
- Output is a set of shards used in an E-mail discovery search application

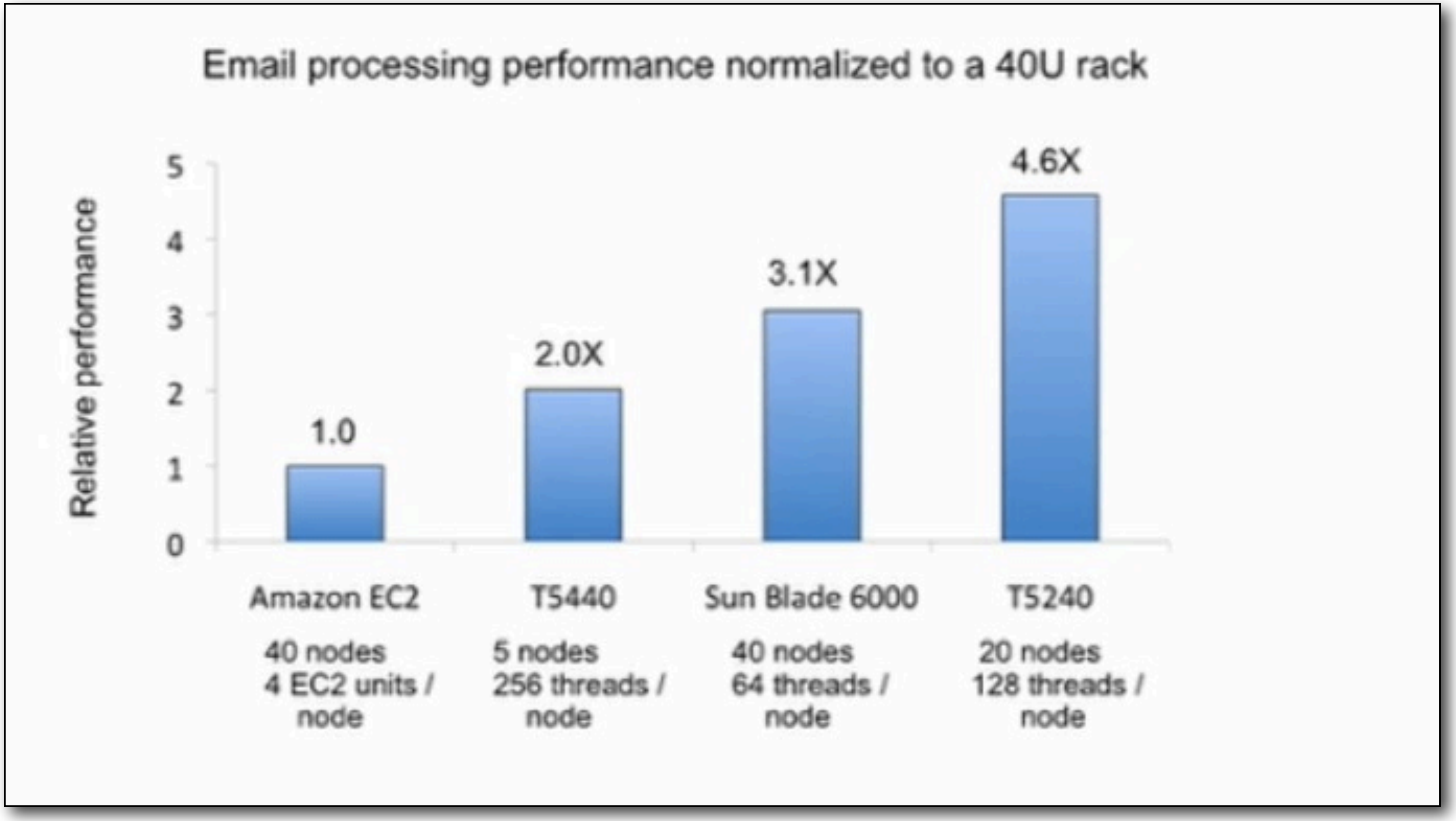
E-mail Discovery (http://www.it-discovery.com/)



E-Discovery Results



CMT Hadoop systems scale for throughput applications


Email processing performance normalized to a 40U rack
Relative performance
0
1
2
3
4
5
1.0
2.0X
3.1X
4.6X
Amazon EC2
40 nodes
4 EC2 units / node
T5440
5 nodes
256 threads / node
Sun Blade 6000
40 nodes
64 threads / node
T5240
20 nodes
128 threads / node

Solaris+Hadoop

I’ve already talked about
Logical Domains and Zones

There is a build-in virtualization in Solaris
It's called Zones.

It‘s an low/no-overhead virtualization

a single kernel
look as several ones.

Thus you have a virtual
operating system in your OS.

Up to 8191.

… you will have no memory before reaching this number.

A Solaris Zone

... can't access the hardware directly

... has it's own root

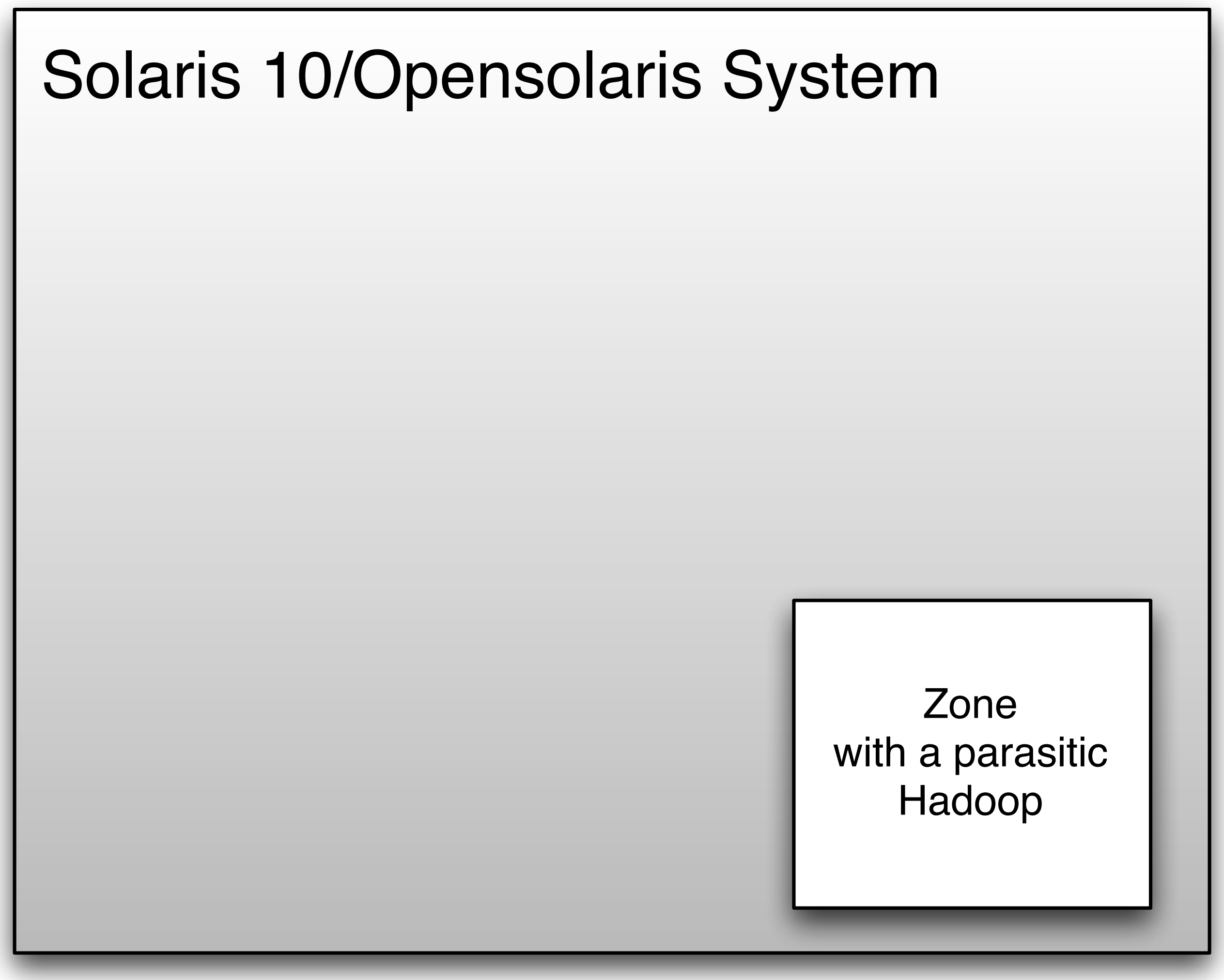
... can't see the contents of other zones

... is a resource management entity

So you could use your normal server systems.

Parasitic Hadoop

It lives from the idle cycles on your systems.


Solaris 10/Opensolaris System
Zone
with a parasitic
Hadoop

A parasite has to ensure that it doesn‘t kill the host, as it would kill the parasite as well.

Solaris has a functionality
called Solaris Resource Management

You can limit the consumption:
... of CPU cycles
... of memory consumption
... of swap space
... of network bandwith

```
#! /usr/bin/perl
while (1) { my $res = ( 3.3333 / 3.14 ) }
```

```
# su einstein
Password:
$ /opt/bombs/cpuhog.pl &
$ /opt/bombs/cpuhog.pl &
```

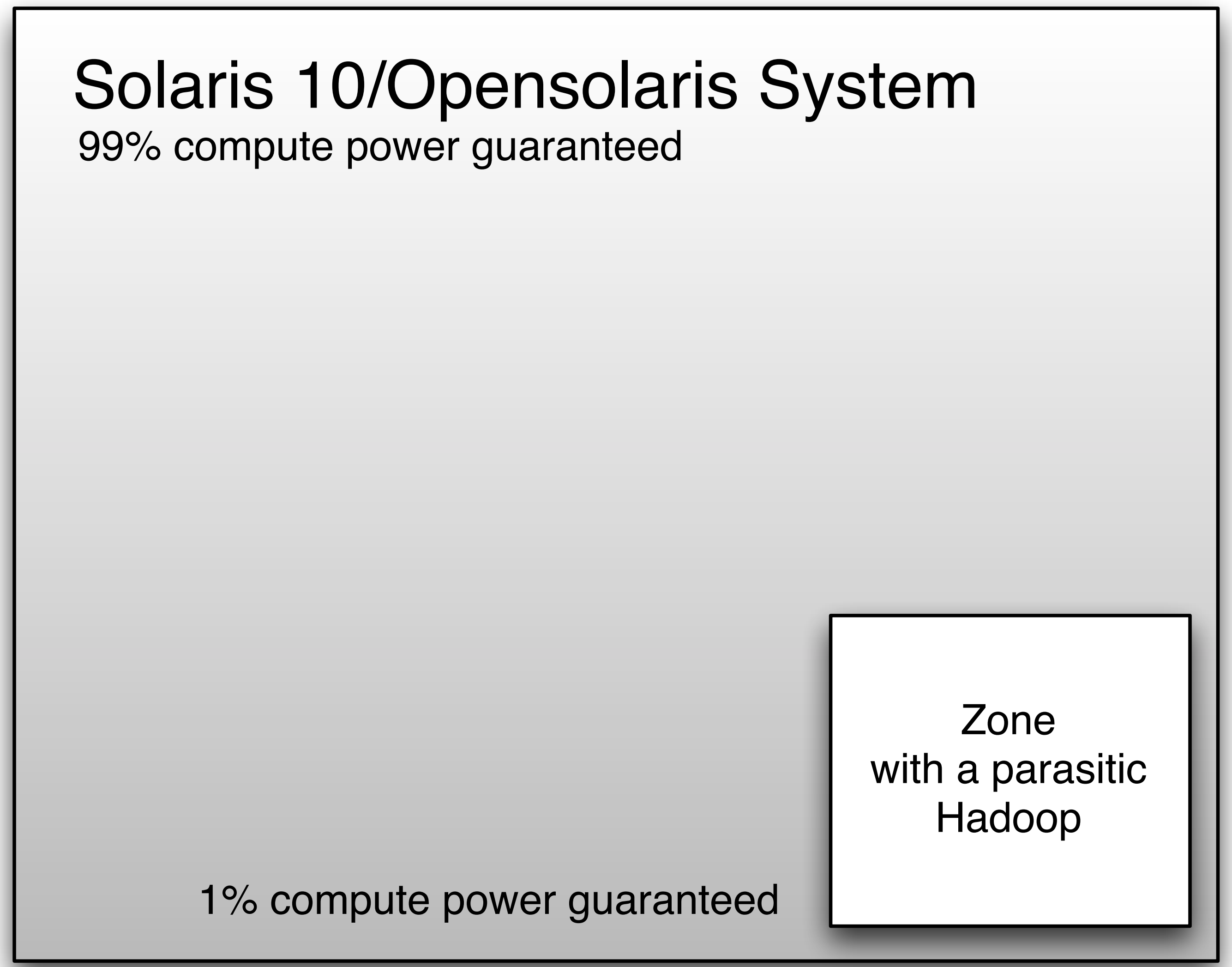
```
bash -3.2$  ps -o pcpu ,project ,args %CPU PROJECT
COMMAND
0.0 user.einstein -sh
0.3 user.einstein bash
47.3 user.einstein /usr/bin/perl /opt/bombs/cpuhog.pl
48.0 user.einstein /usr/bin/perl /opt/bombs/cpuhog.pl
0.2 user.einstein ps -o pcpu,project,args
```

```
# dispadmin -d FSS
```

```
# projadd shcproject
# projmod -U einstein shcproject

# projmod -K  "project.cpu-shares=(privileged ,150,none)" lhcproject
# projmod -K  "project.cpu-shares=(privileged ,50,none)" shcproject
```

```
$ newtask -p shcproject /opt/bombs/cpuhog.pl &

$ ps  -o  pcpu ,project ,args
%CPU PROJECT COMMAND
0.0 user.einstein -sh
0.3 user.einstein bash
0.2 user.einstein ps -o pcpu,project,args
95.9 shcproject /usr/bin/perl /opt/bombs/cpuhog.pl
```

```
$ newtask -p  lhcproject  /opt/bombs/cpuhog.pl   &
[2] 784

$ ps  -o  pcpu ,project ,args
%CPU PROJECT COMMAND
0.0 user.einstein -sh
0.1 user.einstein bash
72.5 lhcproject /usr/bin/perl /opt/bombs/cpuhog.pl
25.6 shcproject /usr/bin/perl /opt/bombs/cpuhog.pl
0.2 user.einstein ps -o pcpu,project,args
```


Solaris 10/Opensolaris System
99% compute power guaranteed
Zone
with a parasitic
Hadoop
1% compute power guaranteed

Icing on the cake

ZFS

Forget everything you know about filesystems:
ZFS isn't really a filesystem ...
A POSIX compatible filesystem is just a possible view
an emulated block device is another ...

No volumes

Integrated RAID

(RAID done right - RAID5/RAID6/RAID TP without read-amplification and write-hole)

Usage-aware selective resilvering

Creating filesystem as easy as directories

Guaranteed data validity (okay 99,9999999999999999999%)

Guaranteed consistent on-disk state of the filesystem

Integrated compression

Integrated Deduplication

More important for our „parasitic Hadoop“:
Quota+Reservations

Putting the HDFS in an own filesystem

Reservation:
ensuring that a filesystem has a certain minimum of free space that can‘t be used by other filesystems

Quota:
ensuring that a filesystem can‘t get bigger than a certain size.

Sun Grid Engine+Hadoop

Great by itself on dedicated machines

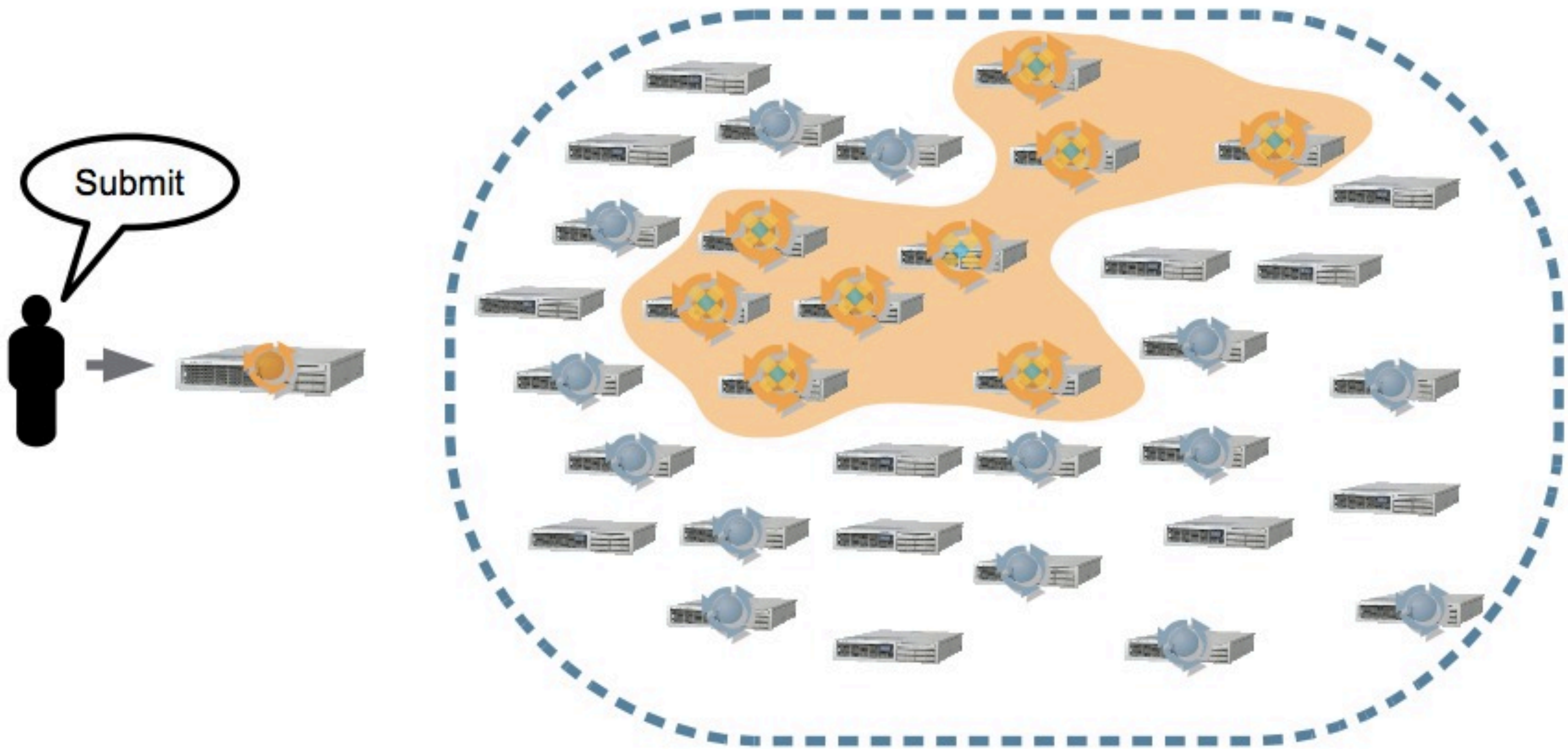
- Map/reduce only
- Unaware of other machine load

Schedules only against data

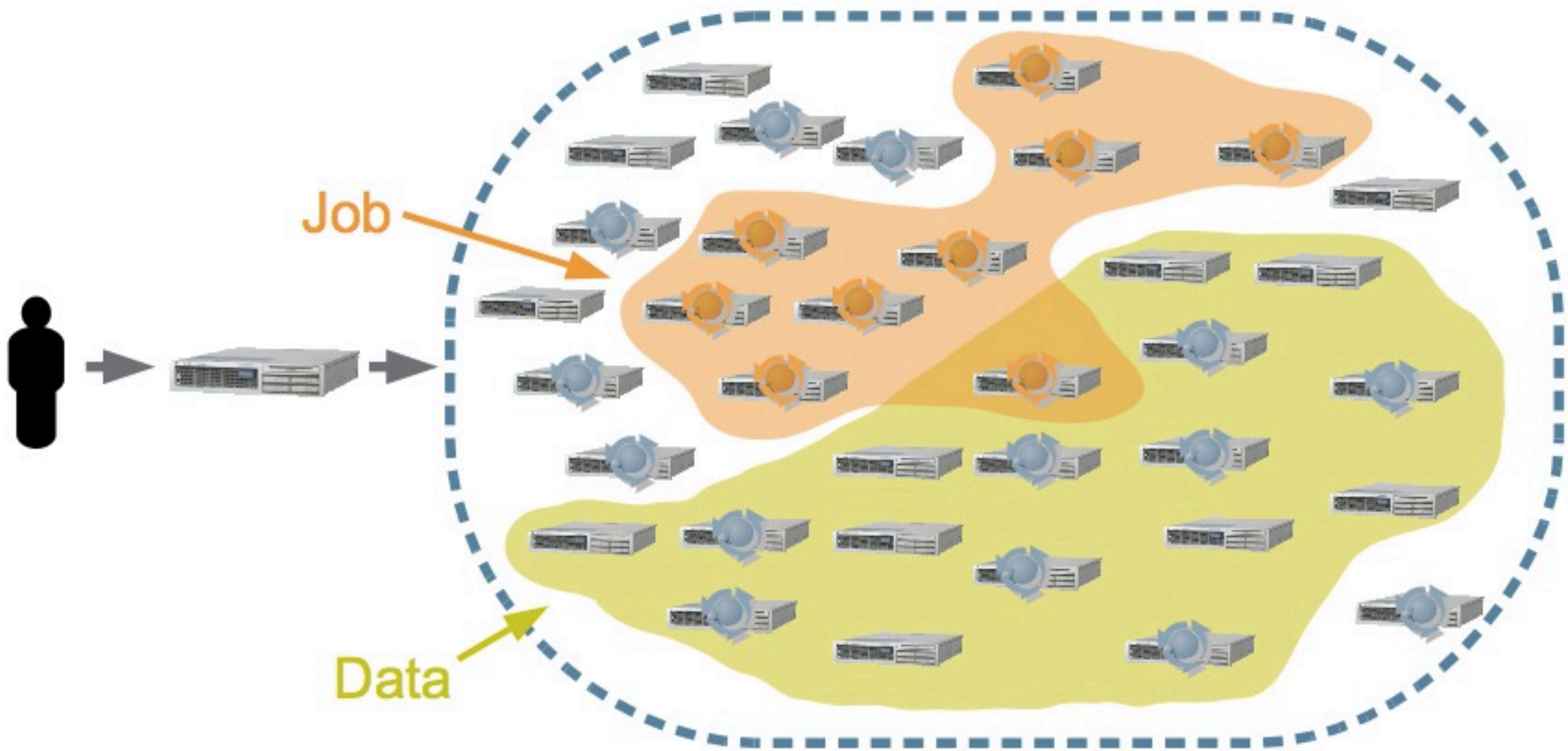
- No policies
- No resource differentiation

No accounting

All things that DRMs do well


Submit

The Hadoop-on-Demand works resonably well but has a problem: It doesn‘t know about the location of the data in the HDFS.



Scheduling Against the Data

Grid Engine resources, aka “complexes”

Model aspects of your cluster

- Concrete
  - Free memory
  - Software licenses
- Abstract
  - High priority
  - Exclusive host access

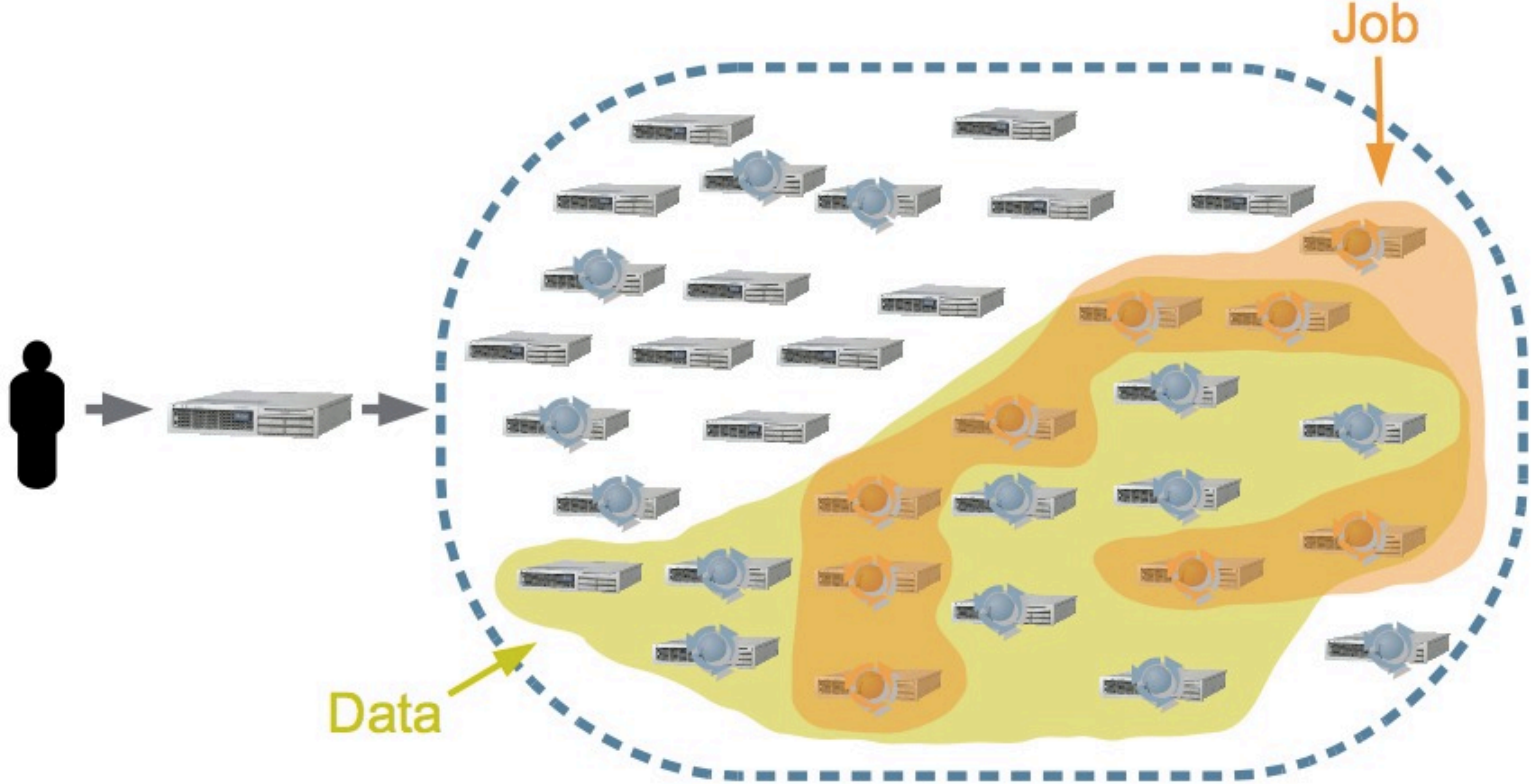
Can be fixed, counted, or *measured*

Why not model HDFS data blocks as resources?

Scheduling Against the Data

The new integreation „measures“ where blocks are ...
... a helper software finds out which blocks you need ...
... and schedules your Hadoop accordingly on this grid nodes.

The new Sun Grid Engine integration of hadoop is data locality aware



Vielen Dank für Ihre Aufmerksamkeit!

Jörg Möllenkamp
Principal Field Technologist

Sun Microsystems