Simplest Scalable Architecture

NOW – Network Of Workstations

Many types of Clusters

(form HP’s Dr. Bruce J. Walker)

- High Performance Clusters
  - Beowulf; 1000 nodes; parallel programs; MPI
- Load-leveling Clusters
  - Move processes around to borrow cycles (eg. Mosix)
- Web-Service Clusters
  - LVS; load-level tcp connections; Web pages and applications
- Storage Clusters
  - parallel filesystems; same view of data from each node
- Database Clusters
  - Oracle Parallel Server;
- High Availability Clusters
  - ServiceGuard, Lifekeeper, Failsafe, heartbeat, failover clusters

Many types of Clusters

(form HP’s Dr. Bruce J. Walker)

- High Performance Clusters
  - Beowulf; 1000 nodes; parallel programs; MPI
- Load-leveling Clusters
  - Move processes around to borrow cycles (eg. Mosix)
- Web-Service Clusters
  - LVS; load-level tcp connections; Web pages and applications
- Storage Clusters
  - parallel filesystems; same view of data from each node
- Database Clusters
  - Oracle Parallel Server;
- High Availability Clusters
  - ServiceGuard, Lifekeeper, Failsafe, heartbeat, failover clusters

NOW type architectures

NOW Approaches

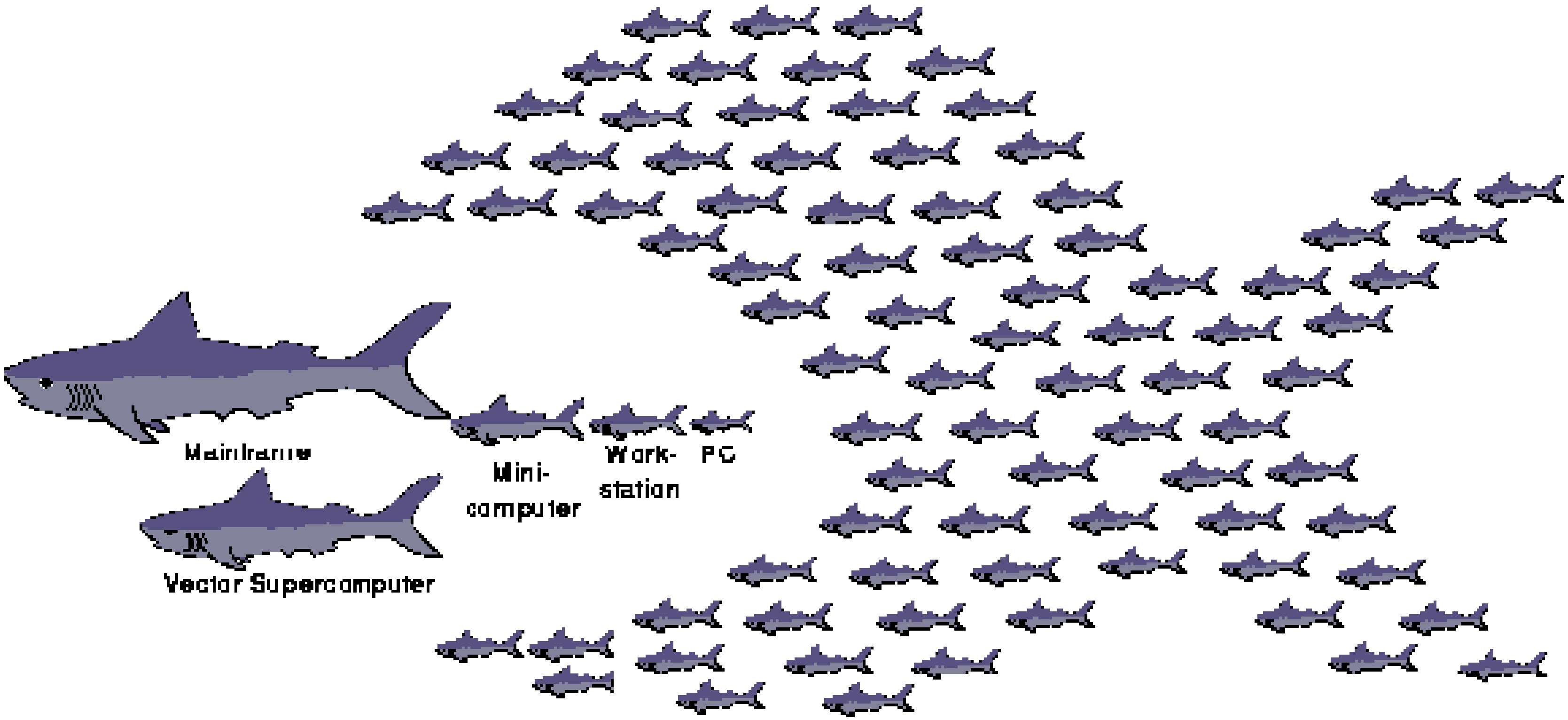
- Single System View
- Shared Resources
- Virtual Machine
- Single Address Space

Shared System View

- Loadbalancing clusters
- High availability clusters
- High Performance
  - High throughput
  - High capability

Cluster Computing

Berkeley NOW



NOW

NOW Philosophies

- Commodity is cheaper
- In 1994 1 MB RAM was
  - $40/MB for a PC
  - $600/MB for a Cray M90

NOW Philosophies

- Commodity is faster

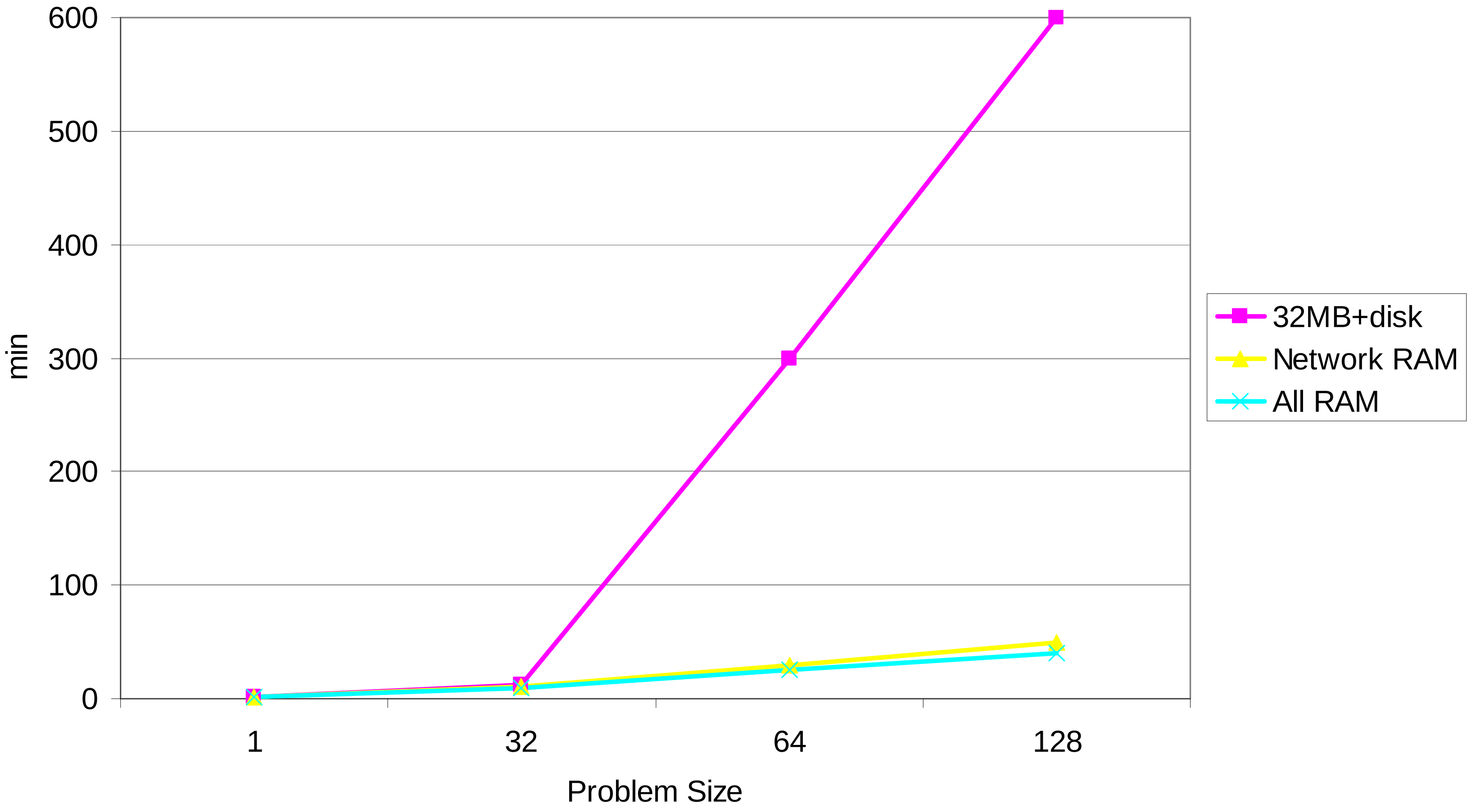
CPU	**MPP year**	**WS year**
150 MHz Alpha	93-94	92-93
50MHz i860	92-93	~91
32 MHz SS-1	91-92	89-90

Network RAM

- Swapping to disk is extremely expensive
  - 16-24 ms for a page swap on disk
- Network performance is much higher
  - 700 us for page swap over the net

Cluster Computing

Network RAM



NOW or SuperComputer?

Machine	**Time**	**Cost**
C-90 (16)	27	$30M
RS6000 (256)	27374	$4M
”+ATM	2211	$5M
”+Parallel FS	205	$5M
”+NOW protocol	21	$5M

NOW Projects

- Condor
- DIPC
- MOSIX
- GLUnix
- PVM
- MUNGI
- Amoeba

The Condor System

- Unix and NT
- Operational since 1986
- More than 1300 CPUs at UW-Madison
- Available on the web
- More than 150 clusters worldwide in academia and industry

What is Condor?

- Condor converts collections of distributively owned workstations and dedicated clusters into a high-throughput computing facility.
- Condor uses matchmaking to make sure that everyone is happy.

What is High-Throughput Computing?

- High-performance: CPU cycles/second under ideal circumstances.
  - "How fast can I run simulation X on this machine?"
- High-throughput: CPU cycles/day (week, month, year?) under non-ideal circumstances.
  - "How many times can I run simulation X in the next month using all available machines?"

What is High-Throughput Computing?

- Condor does whatever it takes to run your jobs, even if some machines…
  - Crash! (or are disconnected)
  - Run out of disk space
  - Don’t have your software installed
  - Are frequently needed by others
  - Are far away & admin’ed by someone else

A Submit Description File

```
# Example condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are not
#       case sensitive, but filenames are!
Universe   = vanilla
Executable = /home/wright/condor/my_job.condor
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
Arguments  = -arg1 -arg2
InitialDir = /home/wright/condor/run_1
Queue
```

What is Matchmaking?

- Condor uses Matchmaking to make sure that work gets done within the constraints of both users and owners.
- Users (jobs) have constraints:
  - “I need an Alpha with 256 MB RAM”
- Owners (machines) have constraints:
  - “Only run jobs when I am away from my desk and never run jobs owned by Bob.”

Process Checkpointing

- Condor's Process Checkpointing mechanism saves all the state of a process into a checkpoint file
  - Memory, CPU, I/O, etc.
- The process can then be restarted *from right where it left off*
- Typically no changes to your job's source code needed – however, your job must be *relinked* with Condor's Standard Universe support library

Remote System Calls

- I/O System calls trapped and sent back to submit machine
- Allows Transparent Migration Across Administrative Domains
  - Checkpoint on machine A, restart on B
- No Source Code changes required
- Language Independent
- Opportunities for Application Steering
  - Example: Condor tells customer process “how” to open files

DIPC

- DIPC
  - Distributed
  - Inter
  - Process
  - Communication
- Provides Sys V IPC in distributed environments (including SHMEM)

MOSIX and its characteristics

- Software that can transform a Linux cluster of x86 based workstations and servers to run almost like an SMP
- Has the ability to distribute and redistribute the processes among the nodes

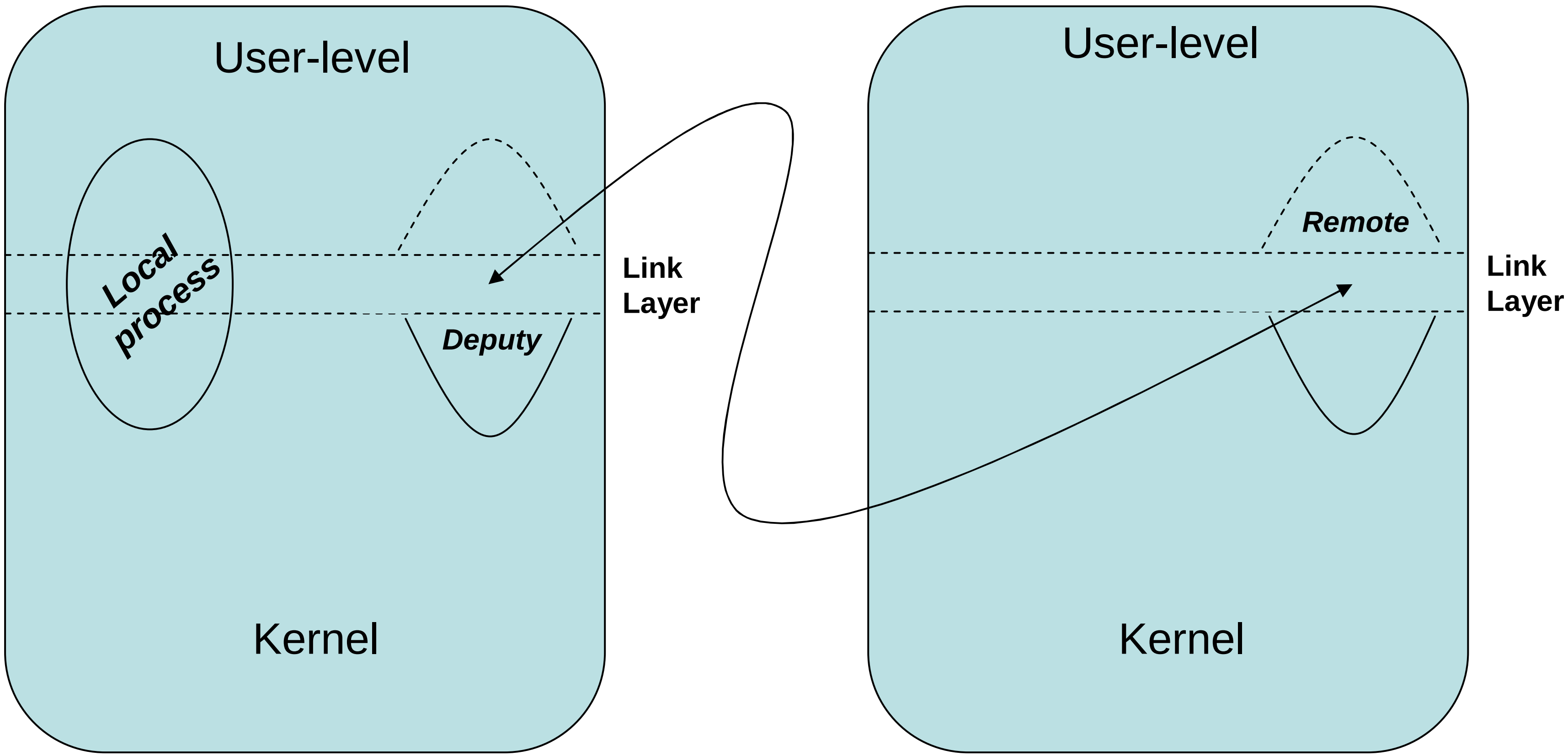
MOSIX

- Dynamic migration added to the BSD kernel
- Uses TCP/IP for communication between workstations
- Requires Homogeneous networks

MOSIX

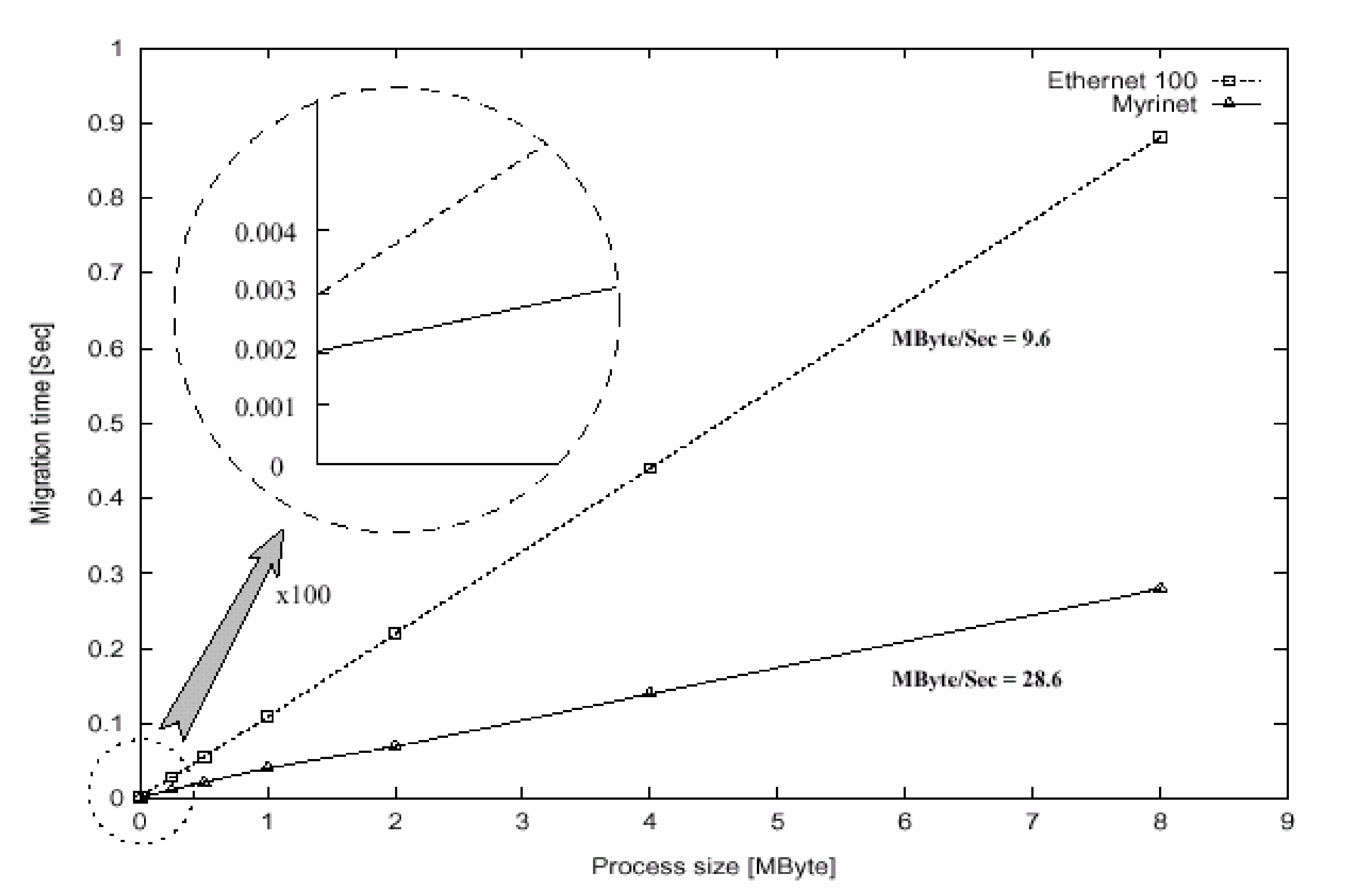
- All processes start their life at the users workstation
- Migration is transparent and preemptive
- Migrated processes use local resources as much as possible and the resources on the home workstation otherwise

Process Migration in MOSIX



A local process and a migrated process

MOSIX



Ethernet 100
Myrinet
MByte/Sec = 9.6
MByte/Sec = 28.6
x100
0.004
0.003
0.002
0.001
0
Migration time [Sec]
Process size [MByte]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9

Mosix Make

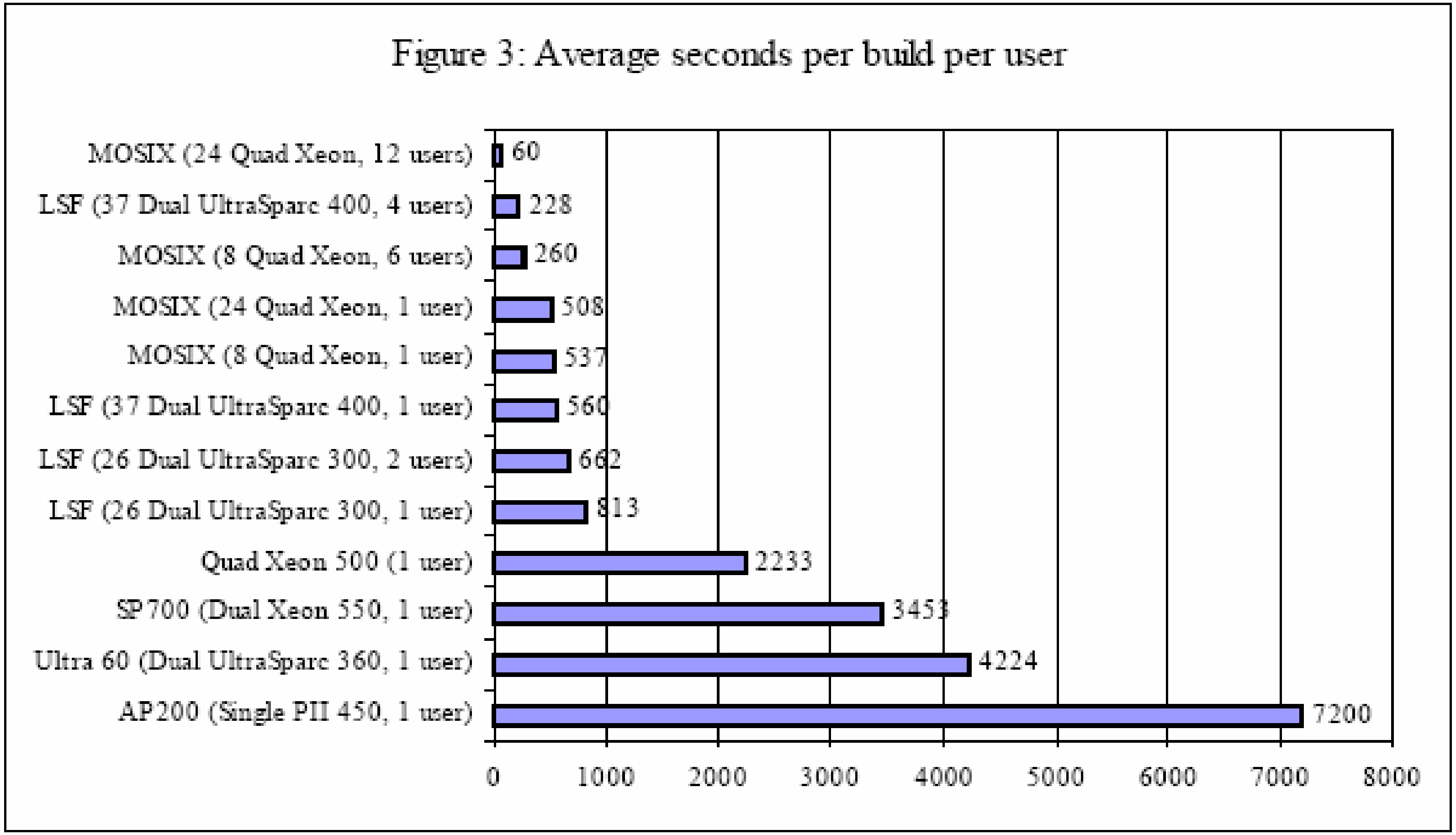


Figure 3: Average seconds per build per user
MOSIX (24 Quad Xeon, 12 users)
LSF (37 Dual UltraSparc 400, 4 users)
MOSIX (8 Quad Xeon, 6 users)
MOSIX (24 Quad Xeon, 1 user)
MOSIX (8 Quad Xeon, 1 user)
LSF (37 Dual UltraSparc 400, 1 user)
LSF (26 Dual UltraSparc 300, 2 users)
LSF (26 Dual UltraSparc 300, 1 user)
Quad Xeon 500 (1 user)
SP700 (Dual Xeon 550, 1 user)
Ultra 60 (Dual UltraSparc 360, 1 user)
AP200 (Single PII 450, 1 user)
60
228
260
508
537
560
662
813
2233
3453
4224
7200
0
1000
2000
3000
4000
5000
6000
7000
8000

GLUnix

- Global Layer Unix
- Pure user-level layer that takes over the role of the operating system from the point of the user
- New processes can then be placed where there is most available memory (CPU)

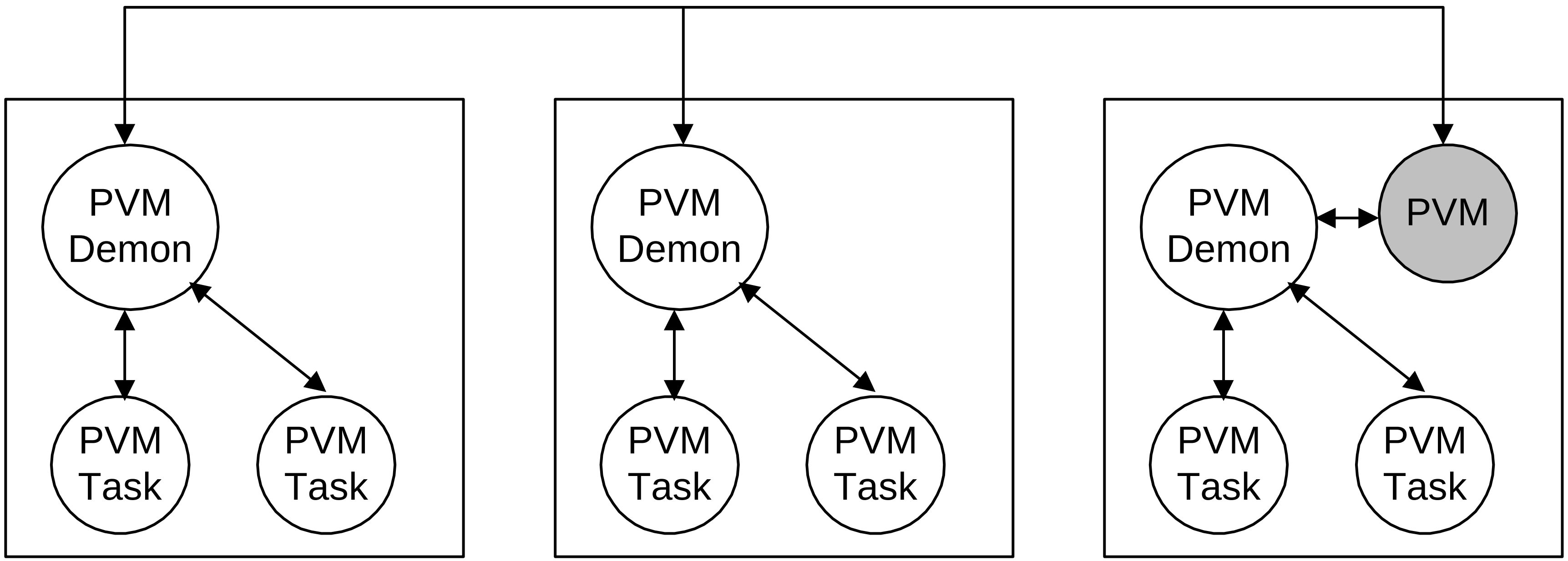
PVM

- Provides a virtual machine on top of existing OS on a network
- Processes can still access the native OS resources
- PVM supports heterogeneous environments!

PVM

- The primary concern of PVM is to provide
  - Dynamic process creation
  - Process management – including signals
  - Communication between processes
  - The machine can be handled during runtime

PVM


PVM
Demon
PVM
Task
PVM
Task
PVM
Demon
PVM
Task
PVM
Task
PVM
Demon
PVM
PVM
Task
PVM
Task

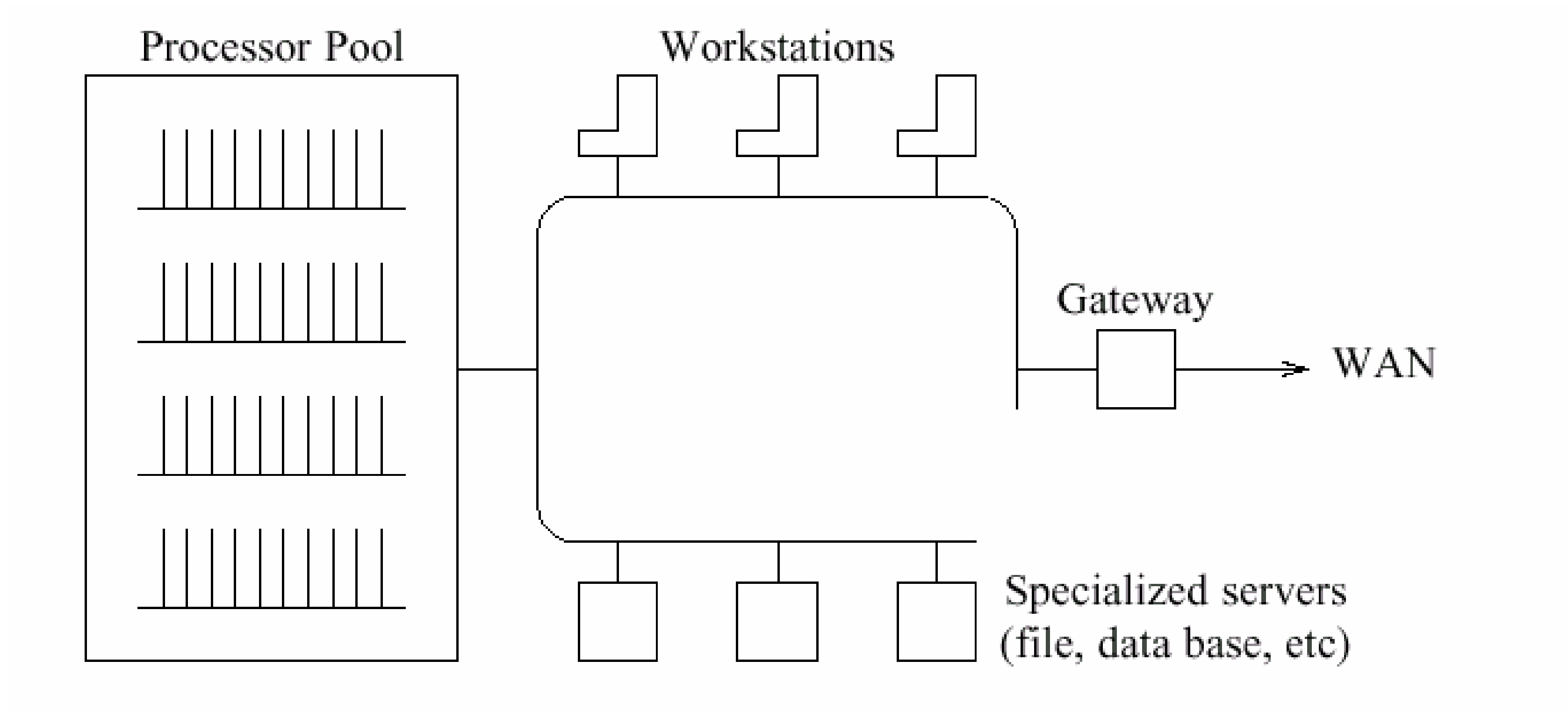
MUNGI

- Single Address Space Operating system
- Requires 64 bit architecture
- Designed as an object based OS
- Protection is implemented as capabilities, to ensure scalability MUNGI uses capability trees rather than lists

Amoeba

- The computer is modeled as a network of resources
- Processes are started where they best fit
- Protection is implemented as capability lists
- Amoeba is centered around an efficient broadcast mechanish

Amoeba


Processor Pool
Workstations
Gateway
WAN
Specialized servers
(file, data base, etc)

Programming NOW

- Dynamic load balancing
- Dynamic orchestration

Dynamic Load Balancing

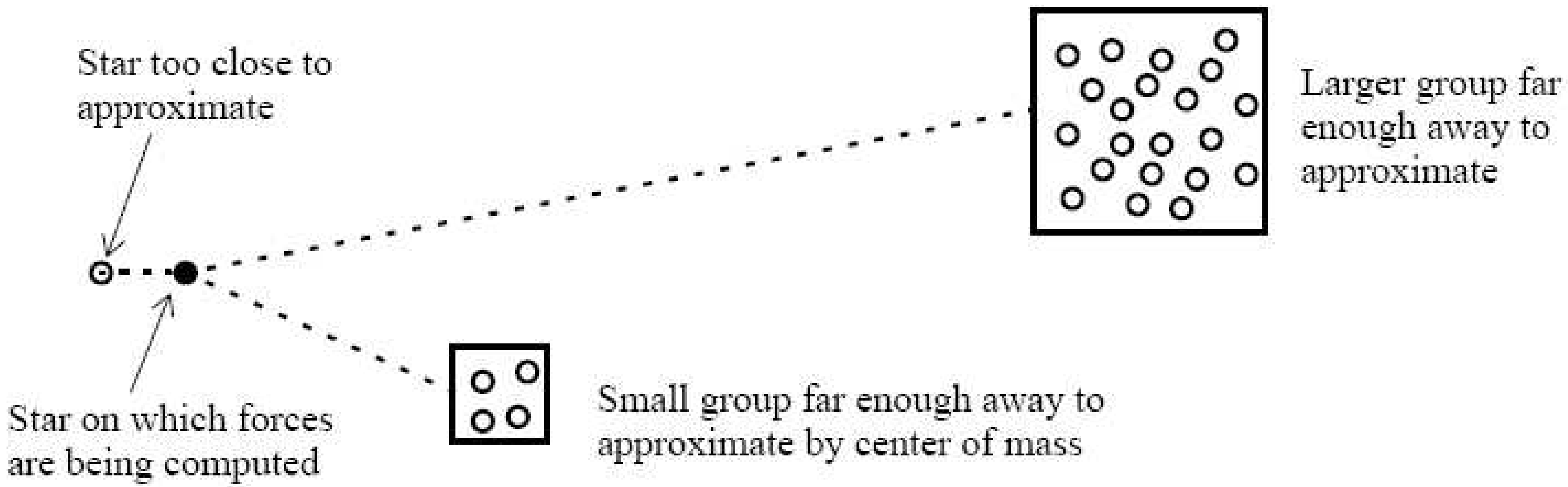
- Base your applications on redundant parallelism
- Rely on the OS to balance the application over the CPUs
- Rather few applications can be orchestrated in this way

Barnes Hut

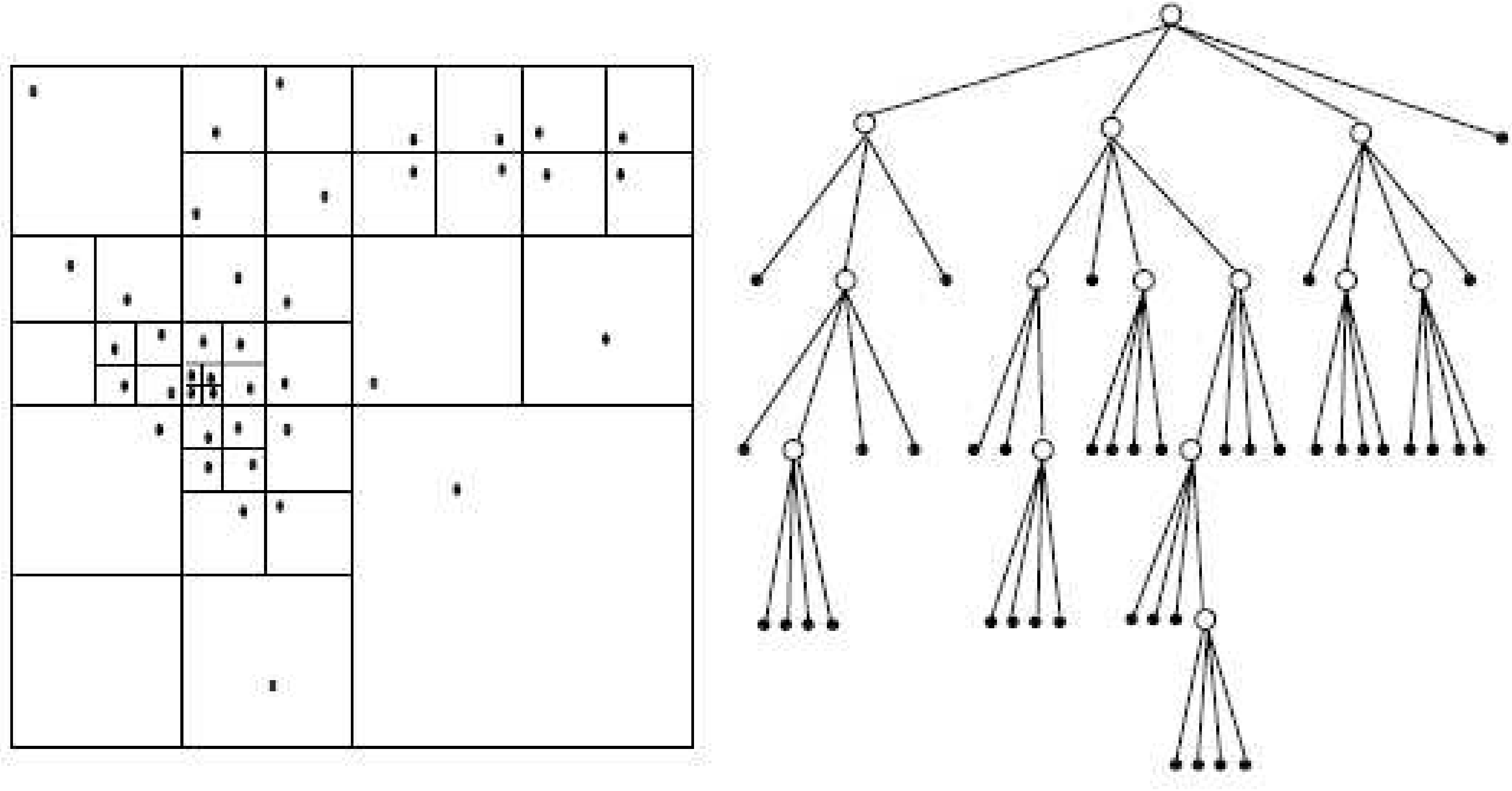
- Galaxy simulations are still quite interresting
- Basic formula is: $G\frac{m_1 m_2}{r^2}$
- Naïve algorithm is O(n^2)

Cluster Computing

Barnes Hut

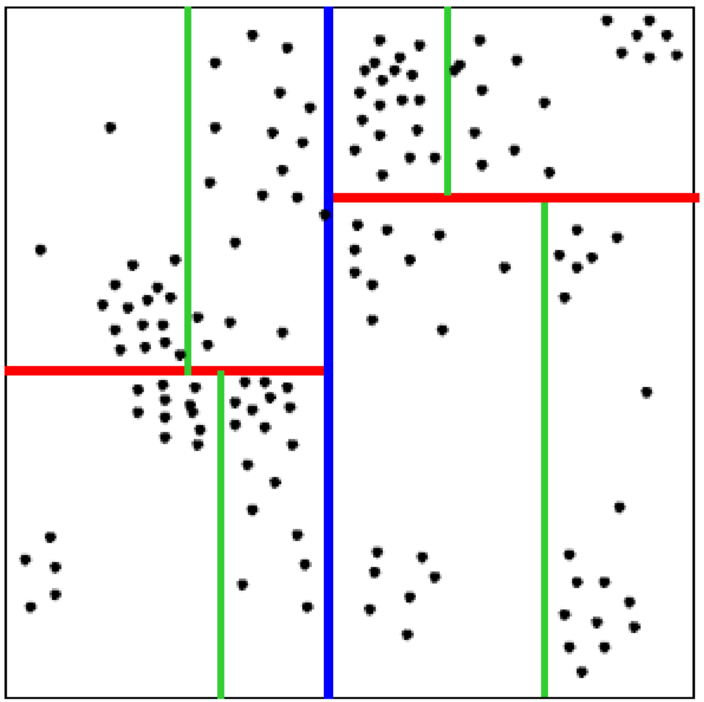


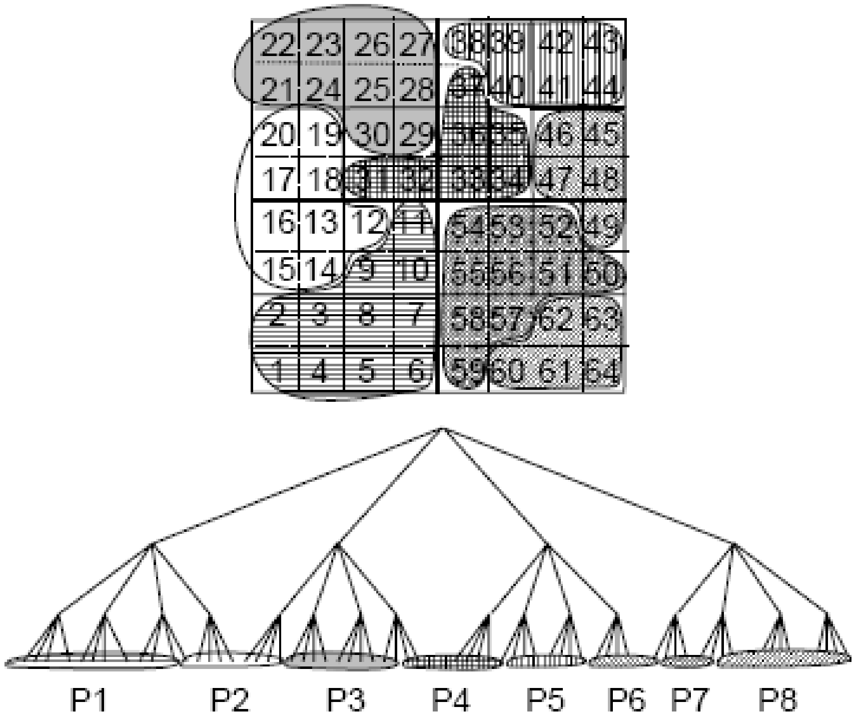
Barnes Hut



O(n log n)

Balancing Barnes Hut





Dynamic Orchestration

- Divide your application into a job-queue
- Spawn workers
- Let the workers take and execute jobs from the queue
- Not all applications can be orchestrated in this way
- Does not scale well – job-queue process may become a bottleneck

Parallel integration

$$\int_{x_1}^{x_2}\int_{y_1}^{y_2}\int_{z_1}^{z_2} f(x,y,z)$$

Parallel integration

- Split the outer integral
- Jobs = range(x_1, x_2, interval)
- Tasks = integral with x_1 = $Jobs_i$, x_2=$Jobs_{i+1}$; for i in len(Jobs -1)
- Result = Sum(Execute(Tasks))