High Throughput Computing (HTC)

Miron Livny
Center for High Throughput Computing
Computer Sciences Department
University of Wisconsin-Madison

The words of Koheleth son of David, king in Jerusalem ~ 200 A.D.

Only that shall happen
Which has happened,
Only that occur
Which has occurred;
There is nothing new
Beneath the sun!

Ecclesiastes Chapter 1 verse 9

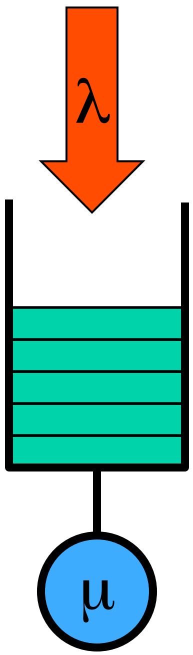
Ecclesiastes, (קֹהֶלֶת, *Kohelet*, "son of David, and king in Jerusalem" alias Solomon, Wood engraving Gustave Doré (1832–1883)

In 1983 I wrote a Ph.D. thesis –

"Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems"

http://www.cs.wisc.edu/condor/doc/livny-dissertation.pdf

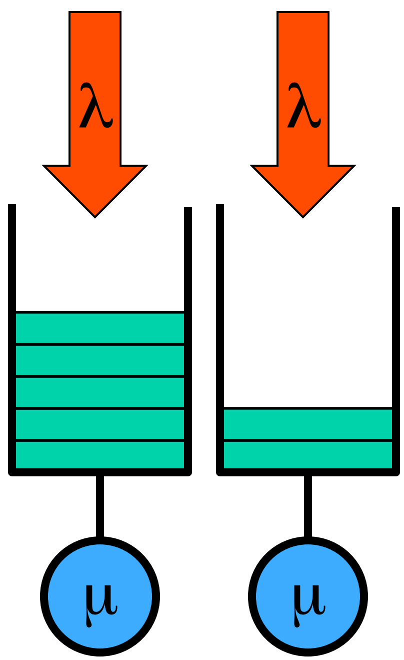
BASICS OF A M/M/1 SYSTEM



Expected # of customers is $1/(1-\rho)$, where ($\rho = \lambda/\mu$) is the utilization

When utilization is 80%, you wait on the average 4 units for every unit of service

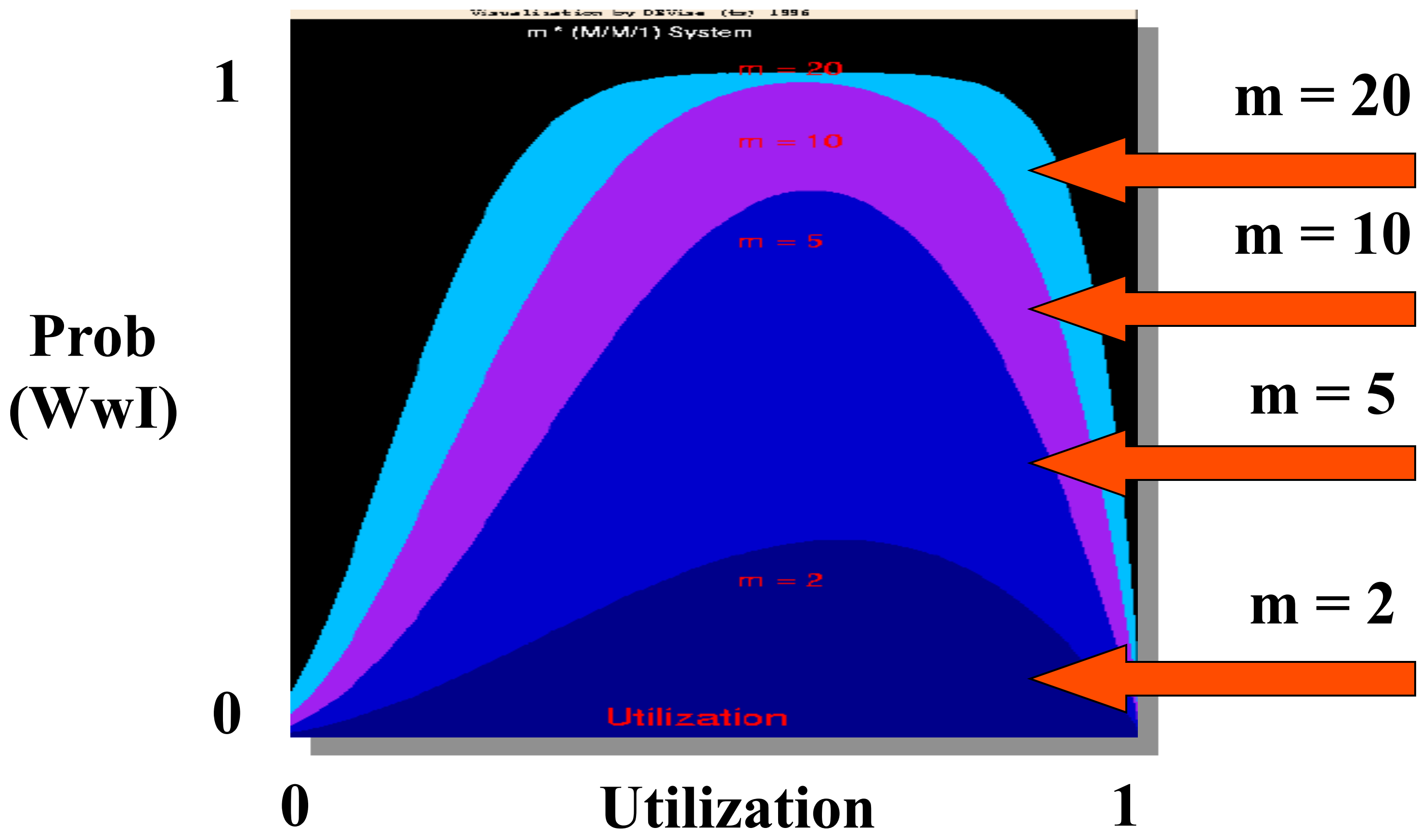
BASICS OF TWO M/M/1 SYSTEMS



When utilization is 80%, you wait on the average 4 units for every unit of service

When utilization is 80%, 25% of the time a customer is waiting for service while a server is idle

Wait while Idle (WwI) in m*M/M/1



Claims for "benefits" provided by Distributed Processing Systems

P.H. Enslow, *"What is a Distributed Data Processing System?"* Computer, January 1978

- High Availability and Reliability
- High System Performance
- Ease of Modular and Incremental Growth
- Automatic Load and Resource Sharing
- Good Response to Temporary Overloads
- Easy Expansion in Capacity and/or Function

Definitional Criteria for a Distributed Processing System

P.H. Enslow and T. G. Saponas *“”Distributed and Decentralized Control in Fully Distributed Processing Systems”* Technical Report, 1981

- Multiplicity of resources
- Component interconnection
- Unity of control
- System transparency
- Component autonomy

Multiplicity of resources

The system should provide a number of assignable resources for any type of **service** demand. The greater the degree of replication of resources, the better the ability of the system to maintain high reliability and performance

Component interconnection

A Distributed System should include a communication subnet which interconnects the elements of the system. The transfer of information via the subnet should be controlled by a two-party, cooperative protocol (**loose coupling**).

Unity of Control

All the component of the system should be **unified** in their desire to achieve a **common goal**. This goal will determine the rules according to which each of these elements will be controlled.

System transparency

From the users point of view the set of resources that constitutes the Distributed Processing System acts like a "**single virtual machine**". When **requesting a service** the user should not require to be aware of the physical location or the instantaneous load of the various resources

Component autonomy

The components of the system, both the logical and physical, should be **autonomous** and are thus afforded the ability to refuse a request of service made by another element. However, in order to achieve the system's goals they have to interact in a **cooperative** manner and thus adhere to a common set of policies. These policies should be carried out by the control schemes of each element.

" … Since the early days of mankind the primary motivation for the establishment of *communities* has been the idea that by being part of an organized group the capabilities of an individual are improved. The great progress in the area of inter-computer communication led to the development of means by which stand-alone processing sub-systems can be integrated into multi-computer '*communities*'. … "

Miron Livny, " *Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems.*",
Ph.D thesis, July 1983.

What Did We Learn From Serving a Quarter of a Million Batch Jobs on a Cluster of Privately Owned Workstations

1992

Miron Livny

Computer Sciences Department
University of Wisconsin — Madison
Madison, Wisconsin
{miron@cs.wisc.edu}

User Prospective

- Maximize the capacity of resources accessible via a single interface
- Minimize overhead of accessing remote capacity
- Preserve local computation environment

High Throughput Computing

We first introduced the distinction between High Performance Computing (HPC) and High Throughput Computing (HTC) in a seminar at the NASA Goddard Flight Center in July of **1996** and a month later at the European Laboratory for Particle Physics (CERN). In June of 1997 HPCWire published an interview on High Throughput Computing.

Why HTC?

For many experimental scientists, scientific progress and quality of research are strongly linked to computing throughput. In other words, they are less concerned about instantaneous computing power. Instead, what matters to them is the amount of computing they can harness over a month or a year --- they measure computing power in units of scenarios per day, wind patterns per week, instructions sets per month, or crystal configurations per year.

High Throughput Computing is a 24-7-365 activity

FLOPY ≠ *(60*60*24*7*52)*FLOPS*

Obstacles to HTC

› Ownership Distribution	(Sociology)
› Customer Awareness	(Education)
› Size and Uncertainties	(Robustness)
› Technology Evolution	(Portability)
› Physical Distribution	(Technology)

Global Scientific Computing via a Flock of Condors

CERN 92

Miron Livny

Computer Sciences Department
University of Wisconsin — Madison
Madison, Wisconsin
{miron@cs.wisc.edu}

MISSION

Give scientists effective and efficient access to large amounts of cheap (if possible free) CPU cycles and main memory storage

THE CHALLENGE

How to turn existing privetly owned clusters of *workstations*, *farms*, *multiprocessors*, and *supercomputers* into an efficient and effective Global Computing Environment?

In other words, how to minimize wait while idle?

APPROACH

Use wide-area networks to transfer batch jobs between Condor systems

- Boundaries of each Condor system will be determined by physical or administrative considerations

TWO EFFORTS

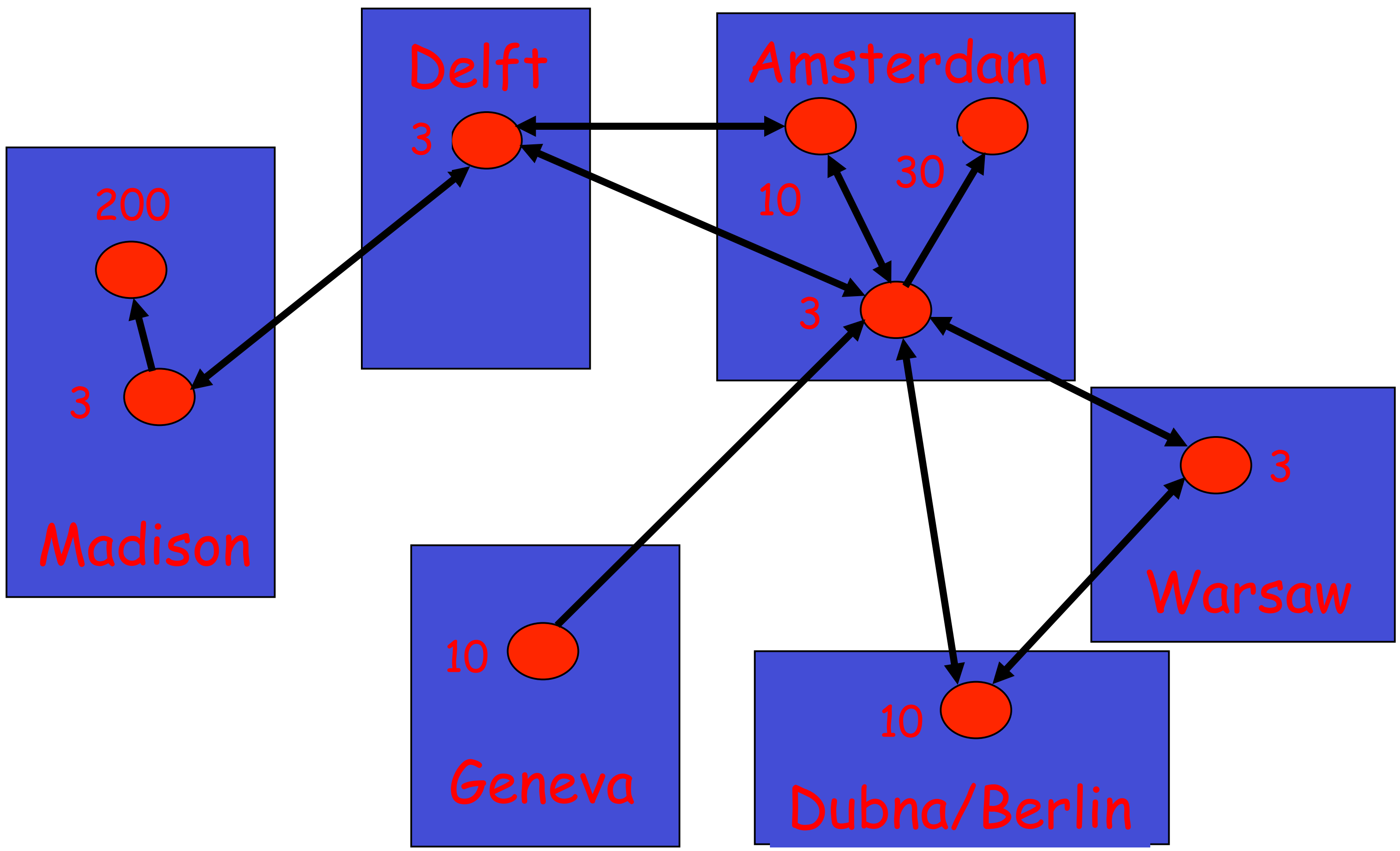
☐ ***UW CAMPUS***

Condor systems at Engineering, Statistics, and Computer Sciences

☐ ***INTERNATIONAL***

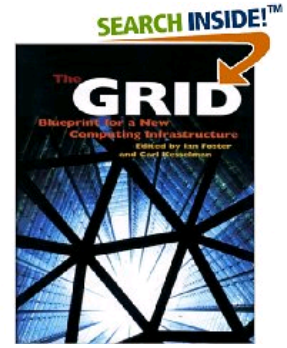
We have started a collaboration between CERN-SMC-NIKHEF-Univ. of Amsterdam, and University of Wisconsin-Madison

The '94 Worldwide Condor Flock



The Grid: Blueprint for a New Computing Infrastructure

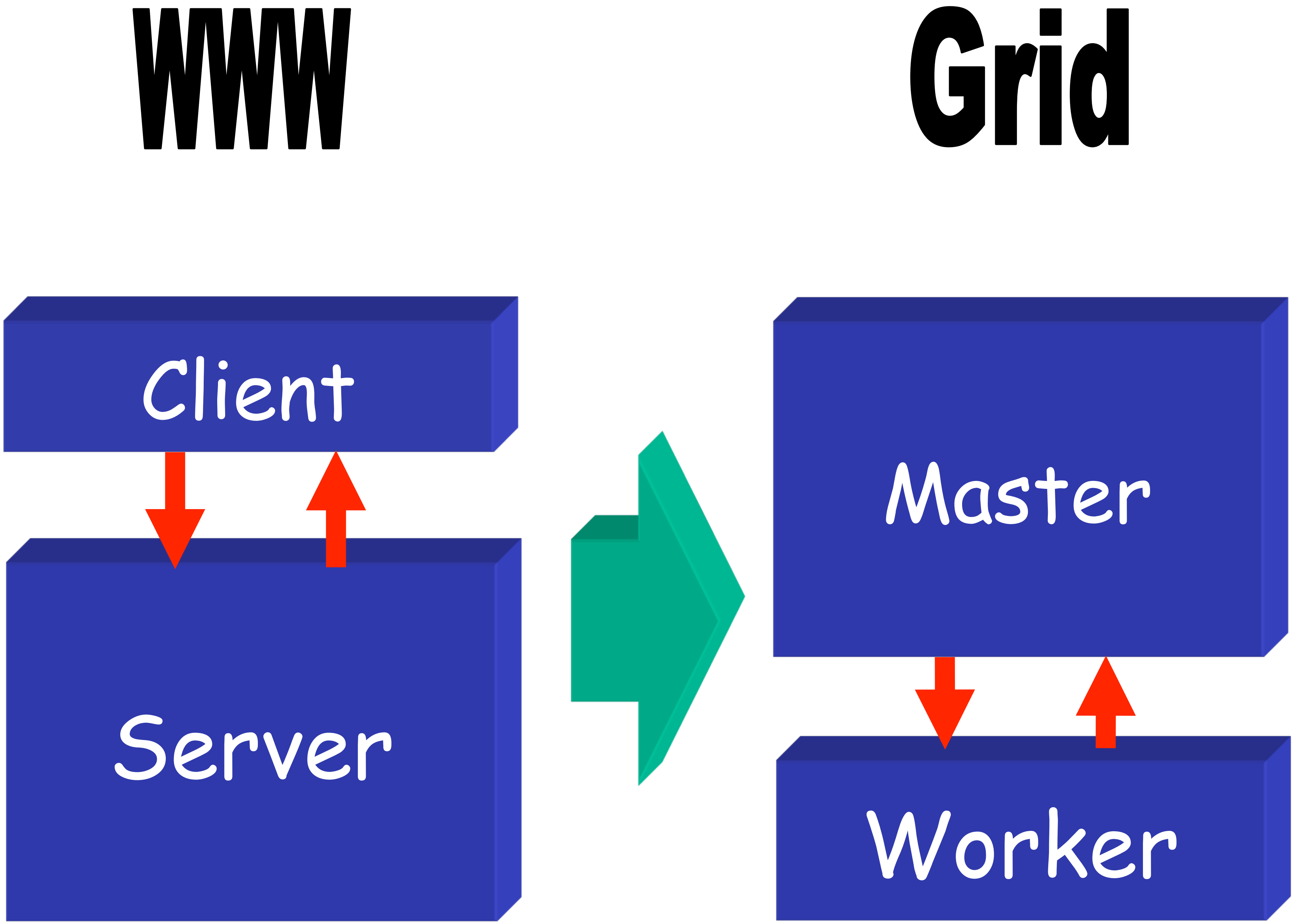
Edited by Ian Foster and Carl Kesselman
July 1998, 701 pages.



The grid promises to fundamentally change the way we think about and use computing. This infrastructure will connect multiple regional and national computational grids, creating a universal source of **pervasive and dependable** computing power that supports dramatically new classes of applications. The Grid provides a clear vision of what computational **grids** are, why we need them, who will use them, and how they will be programmed.

" … We claim that these **mechanisms**, although originally developed in the context of a cluster of workstations, are also applicable to computational **grids**. In addition to the required flexibility of services in these grids, a very important concern is that the system be **robust** enough to run in "**production mode**" continuously even in the face of component failures. … "

Miron Livny & Rajesh Raman, "*High Throughput Resource Management*", **in "*The Grid: Blueprint for a New Computing Infrastructure*".**



WWW
Grid
Client
Server
Master
Worker

The NUG30 Quadratic Assignment Problem (QAP)

$$\min_{p \in \Pi} \sum_{i=1}^{30} \sum_{j=1}^{30} a_{ij} b_{p(i)p(j)}$$

NUG30 Personal Grid …

Managed by **one** Linux box at Wisconsin

Flocking:
- -- the main Condor pool at Wisconsin (500 processors)
- -- the Condor pool at Georgia Tech (284 Linux boxes)
- -- the Condor pool at UNM (40 processors)
- -- the Condor pool at Columbia (16 processors)
- -- the Condor pool at Northwestern (12 processors)
- -- the Condor pool at NCSA (65 processors)
- -- the Condor pool at INFN Italy (54 processors)

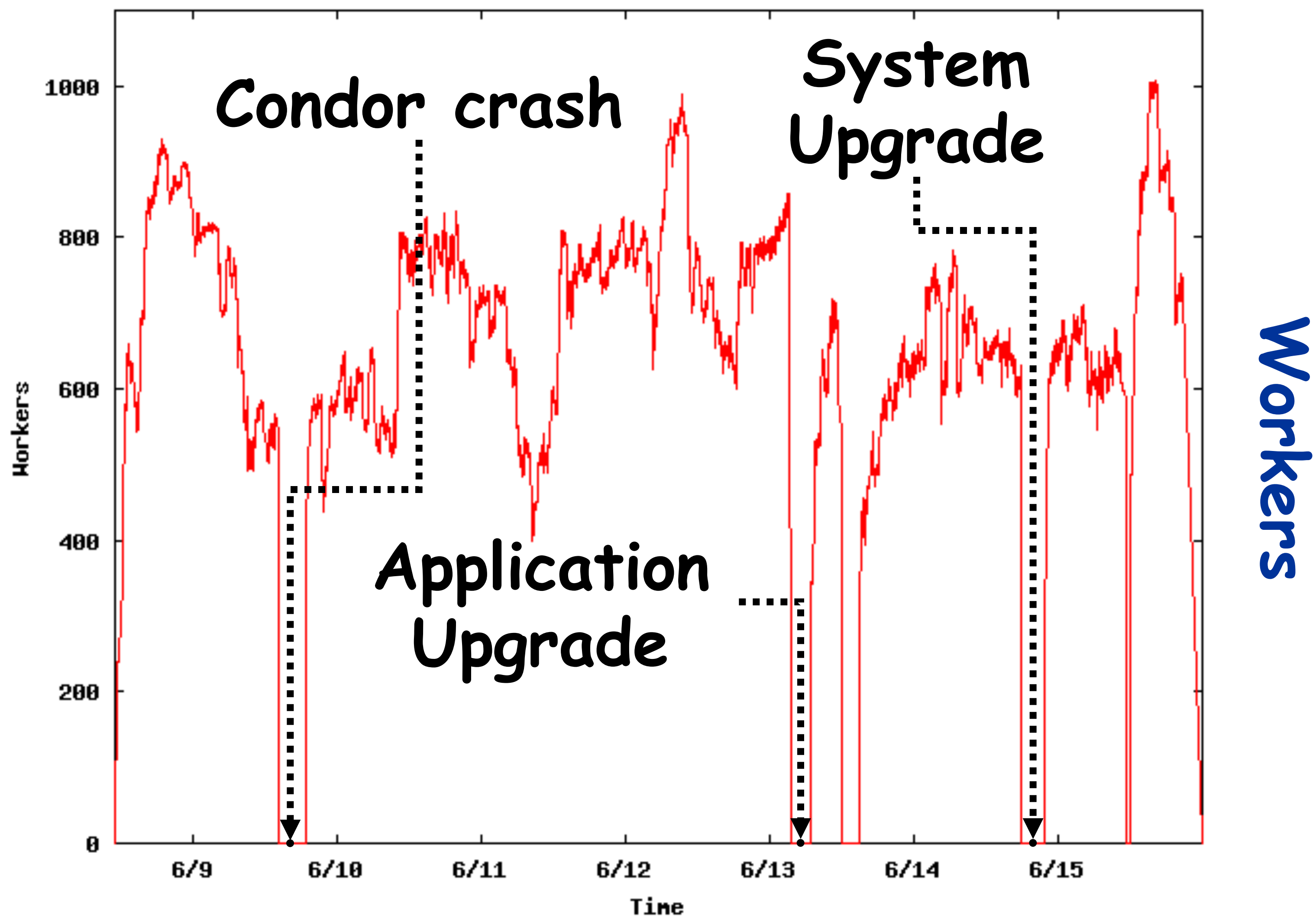
Glide-in:
- -- Origin 2000 (through LSF) at NCSA. (512 processors)
- -- Origin 2000 (through LSF) at Argonne (96 processors)

Hobble-in:
- -- Chiba City Linux cluster (through PBS) at Argonne (414 processors).

Solution Characteristics.

Scientists	**4**
Workstations	**1**
Wall Clock Time	6:22:04:31
Avg. # CPUs	653
Max. # CPUs	1007
Total CPU Time	Approx. 11 years
Nodes	11,892,208,412
LAPs	574,254,156,532
Parallel Efficiency	92%

The NUG30 Workforce



Being a Master

Customer "delegates" task(s) to the master who is responsible for:

* Obtaining **allocation** of resources
* Deploying and managing workers on allocated resources
* **Delegating** work unites to deployed workers
* Receiving and processing results
* Delivering results to customer

Master must be …

- Persistent – work and results must be safely recorded on non-volatile media
- Resourceful – delegates "DAGs" of work to other masters
- Speculative – takes chances and knows how to recover from failure
- Self aware – knows its own capabilities and limitations
- Obedience – manages work according to plan
- Reliable – can mange "large" numbers of work items and resource providers
- Portable – can be deployed "on the fly" to act as a "sub master"

“ … Grid computing is a partnership between clients and servers. Grid clients have more responsibilities than traditional clients, and must be equipped with powerful mechanisms for dealing with and recovering from failures, whether they occur in the context of remote execution, work management, or data output. When clients are powerful, servers must accommodate them by using careful protocols.… “

Douglas Thain & Miron Livny, *"Building Reliable Clients and Servers"*, ***in “The Grid: Blueprint for a New Computing Infrastructure”,2nd edition***

Resource Allocation
(resource -> customer)

vs.

Work Delegation
(job -> resource)


Delegate work
Delegate work
W
R
W
R
Allocate Resource
Allocate Resource

Resource Allocation

A limited assignment of temporary "ownership" of a resource offered by a provider to a requestor

- Requestor is charged for allocation regardless of actual consumption
- Requestor may be given the right to allocate resource to others
- Provider has the right and means to revoke the allocation
- Allocation is governed by an "agreement" between the provider and the requestor
- Allocation is a "lease" that expires if not renewed by the requestor
- Tree of allocations

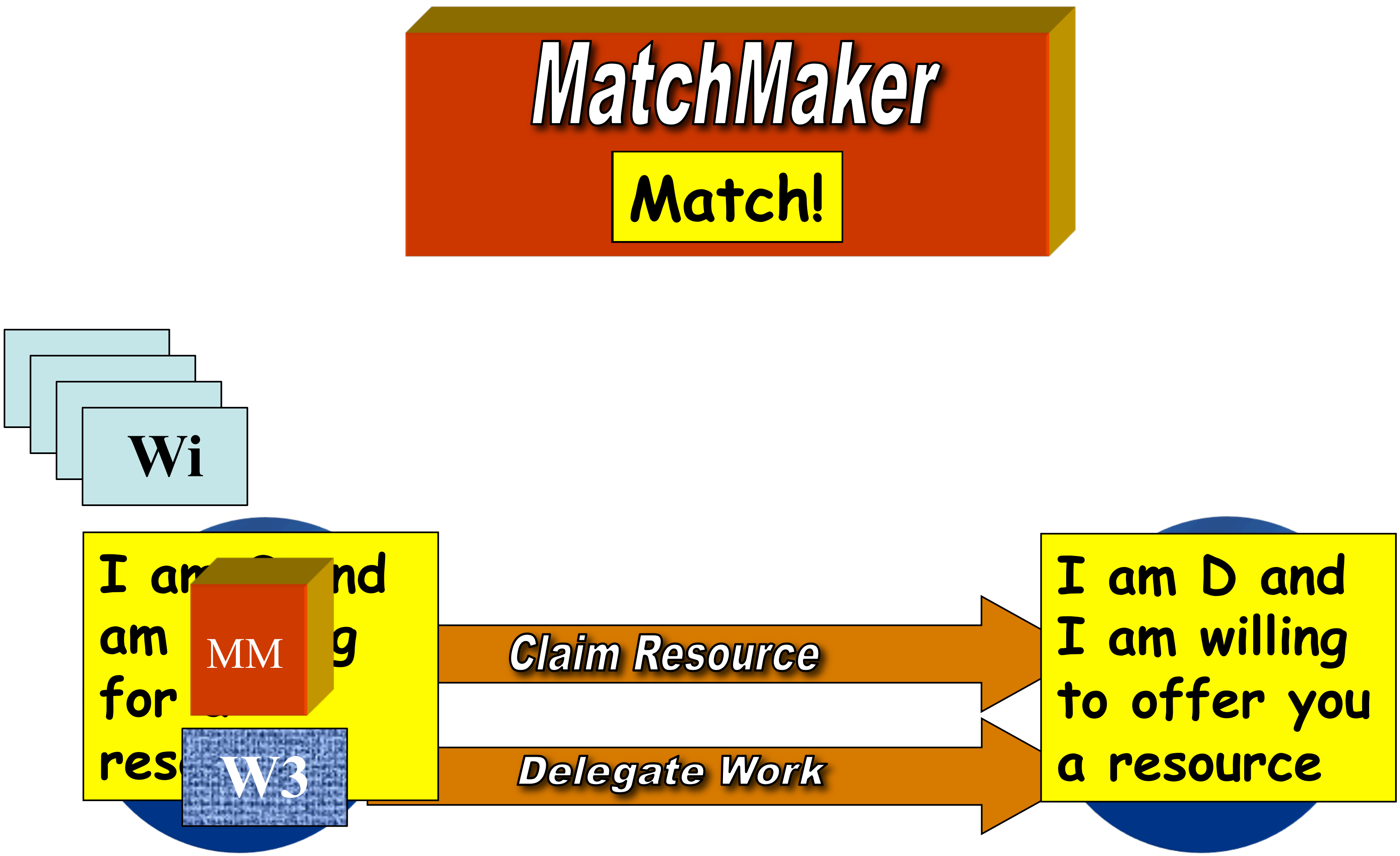
Work Delegation

An assignment of a responsibility to perform a task

- Delegation involved a definition of these “responsibilities”
- Responsibilities my be further delegated
- Delegation consumes resources
- Delegation is a “lease” that expires if not renewed by the assigner
- Can form a tree of delegations

Focus of the grid "movement" has been remote job delegation (Gate Keepers), commercial clouds are all about remote resource allocation (Provisioning)

In Condor we use a two phase matchmaking process to first allocate a resource to a requestor and then to select a task to be delegated to this resource

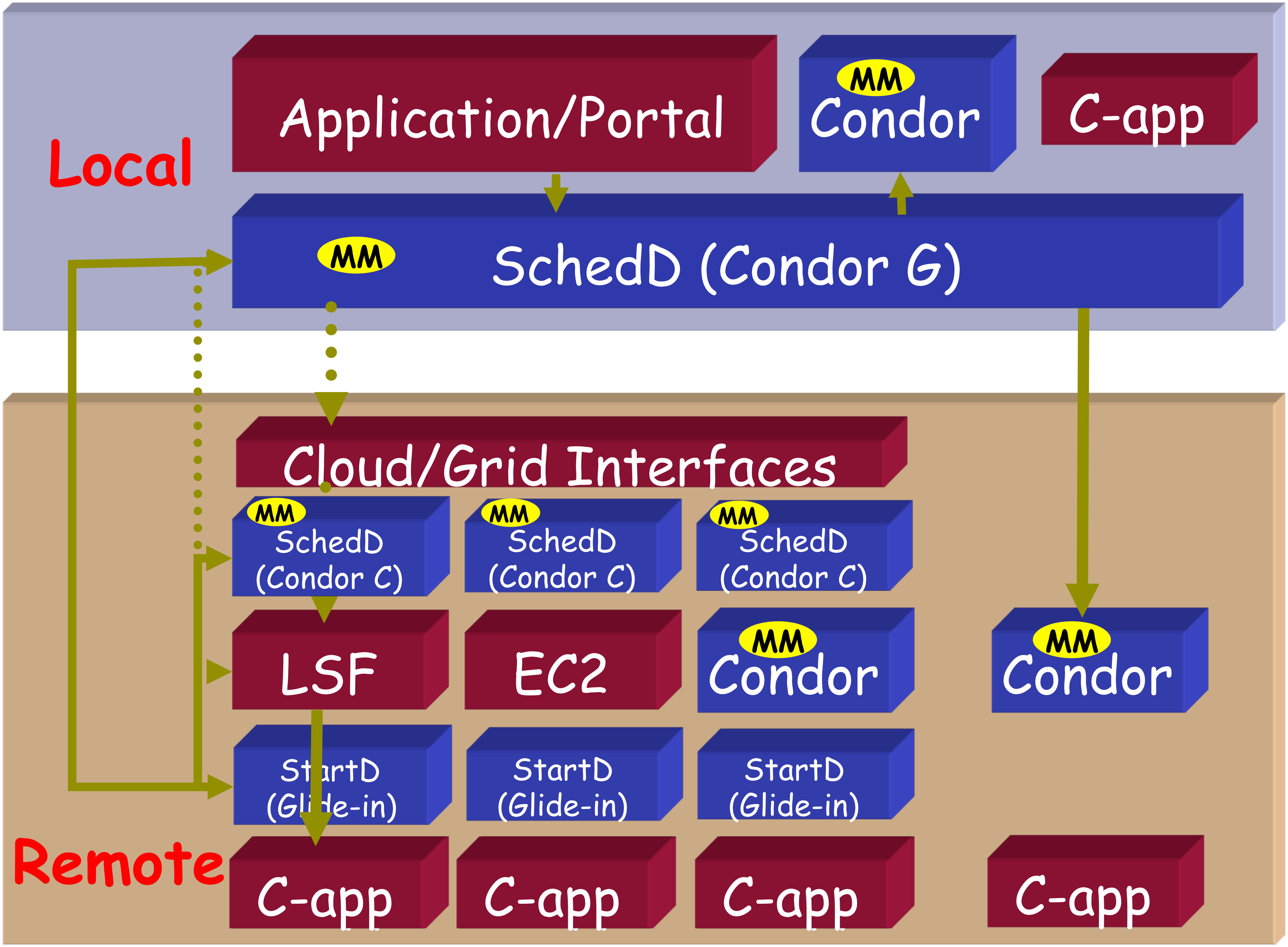


MatchMaker
Match!
Wi
MM
W3
Claim Resource
Delegate Work
I am D and
I am willing
to offer you
a resource

Overlay Resource Managers

Ten years ago we introduced the concept of **Condor glide-ins** as a tool to support 'just in time scheduling' in a distributed computing infrastructure that consists of recourses that are managed by (heterogeneous) autonomous resource managers. By dynamically deploying a distributed resource manager on resources allocated (provisioned) by the local resource managers, the overlay resource manager can implement a unified resource allocation policy.

In other words, we use remote job invocation to get resources allocated.


Local
Application/Portal
MM
Condor
C-app
MM
SchedD (Condor G)
Remote
Cloud/Grid Interfaces
MM
SchedD (Condor C)
MM
SchedD (Condor C)
MM
SchedD (Condor C)
LSF
EC2
MM
Condor
MM
Condor
StartD (Glide-in)
StartD (Glide-in)
StartD (Glide-in)
C-app
C-app
C-app
C-app

How can we accommodate
an **unbounded**
need for computing and
an **unbounded**
amount of data with
an **unbounded**
amount of resources?