The Principles of HTC

Miron Livny
Wisconsin Institutes for Discovery
University of Wisconsin-Madison

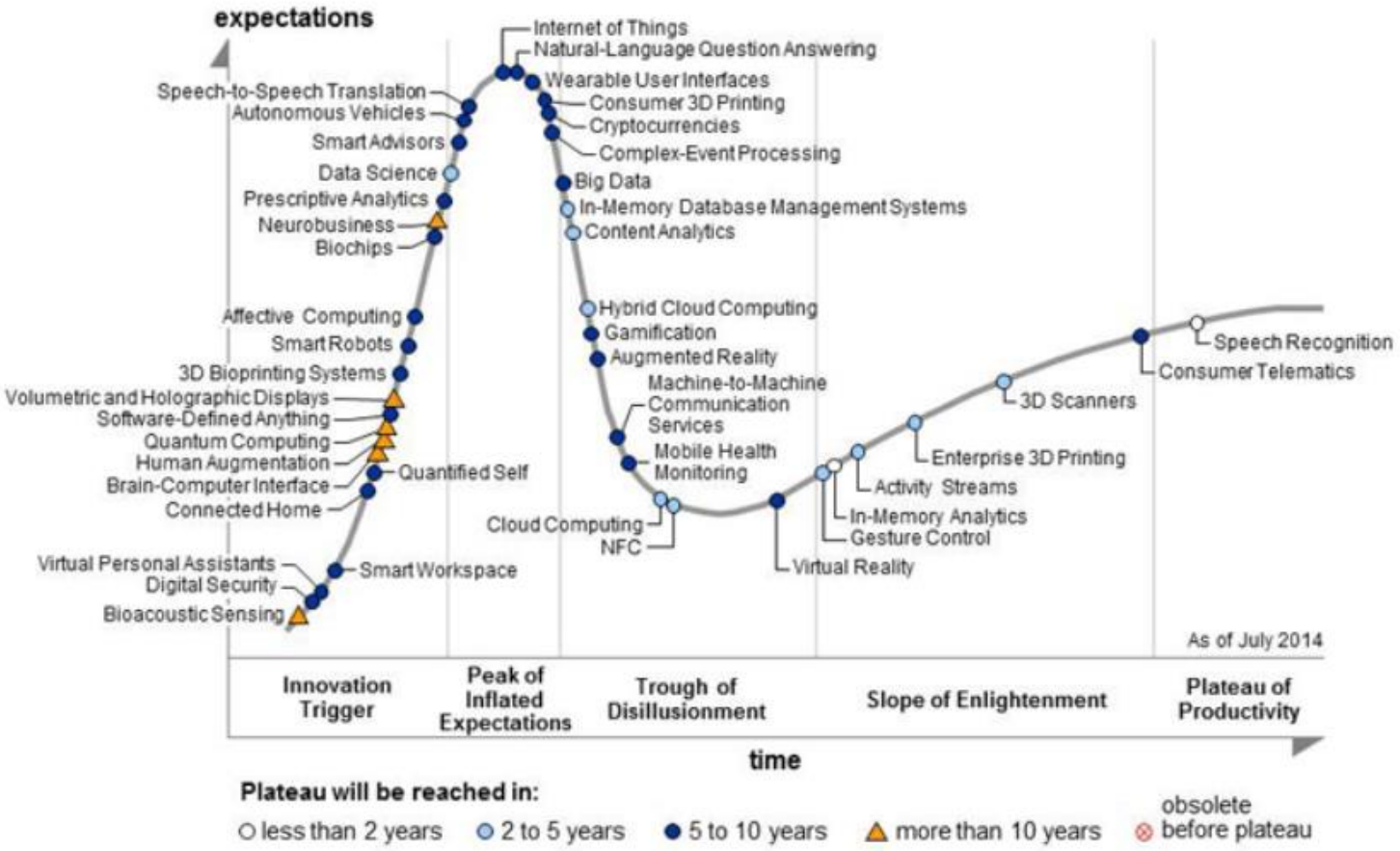
The words of Koheleth son of David, king in Jerusalem ~ 200 A.D.

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

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

Ecclesiastes Chapter 1 verse 9

We are driven by Principals (– Hype)



Source: Gartner (August 2014)

Perspectives on Grid Computing

(2010)

Uwe Schwiegelshohn Rosa M. Badia Marian Bubak Marco Danelutto Schahram Dustdar Fabrizio Gagliardi Alfred Geiger Ladislav Hluchy Dieter Kranzlmüller Erwin Laure Thierry Priol Alexander Reinefeld Michael Resch Andreas Reuter Otto Rienhoff Thomas Rüter Peter Sloot Domenico Talia Klaus Ullmann Ramin Yahyapour Gabriele von Voigt

We should not waste our time in redefining terms or key technologies: clusters, Grids, Clouds... What is in a name? Ian Foster recently quoted Miron Livny saying: "I was doing Cloud computing way before people called it Grid computing", referring to the ground breaking Condor technology. It is the Grid scientific paradigm that counts!

The paradigm shift of 70's – computing hardware packaged and sold in small units

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**

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.

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.

It is always a tradeoff

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

Minimize wait (job/task queued) while Idle (a resource that is capable and willing to serve the job/task is running a lower priority job/task)

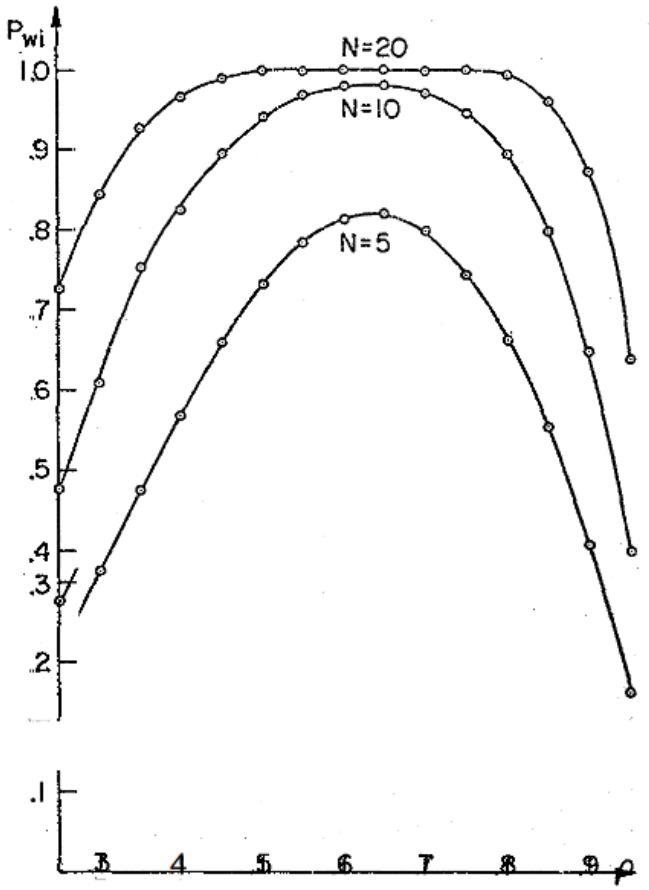


figure: 1 P_{wi} as a function of ρ

When each resource has a it's own queue, when should I stay at the current queue and wait and when should I move to another queue?

" … 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.

In 1985 we extended the scope of the distributed load balancing problem to include "ownership" of resources

Should I share my resource and if I do with whom and when?


MINE
YOURS
OURS

Now you have a community of customers who are consumers, providers or both

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

Submit locally (queue and manage your jobs/tasks locally; leverage your local resources) **and run globally** (acquire any resource that is capable and willing to run your job/task)

- **Job owner identity is local**
  - Owner identity should never “travel” with the job to execution site
  - Owner attributes are local
- **Name spaces are local**
  - File names are locally defined
- **Resource acquisition is local**
  - Submission site (local) is responsible for the acquisition of all resources

- **“external” forces moved us away from this “pure” local centric view of the distributed computing environment.**
- **With the help of capabilities (short lived tokens) and reassignment of responsibilities we are committed to regain full local control.**
- **Handing users with money (real or funny) to acquire commuting resources helps us move (push) in this positive direction.**

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

HIGH THROUGHPUT COMPUTING: AN INTERVIEW WITH MIRON LIVNY 06.27.97
by Alan Beck, editor in chief HPCwire
==========

This month, NCSA's (National Center for Supercomputing Applications) Advanced Computing Group (ACG) will begin testing Condor, a software system developed at the University of Wisconsin that promises to expand computing capabilities through efficient capture of cycles on idle machines. The software, operating within an HTC (High Throughput Computing) rather than a traditional HPC (High Performance Computing) paradigm, organizes machines

Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017-2020

AUTHORS

Committee on Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science in 2017-2020; Computer Science and Telecommunications Board; Division on Engineering and Physical Sciences; National Academies of Sciences, Engineering, and Medicine

"… many fields today rely on **high-throughput computing** for **discovery**."

"Many fields **increasingly** rely **on high-throughput computing**"

High Throughput Computing requires automation as it is a 24-7-365 activity that involves large numbers of jobs

FLOPY ≠ *(60*60*24*7*52)*FLOPS*

100K Hours*1 Job ≠ 1 H*100K J

Using Directed Acyclic Graphs (DAGs) to support declarative automation of interdependent tasks

Courtesy SXS.

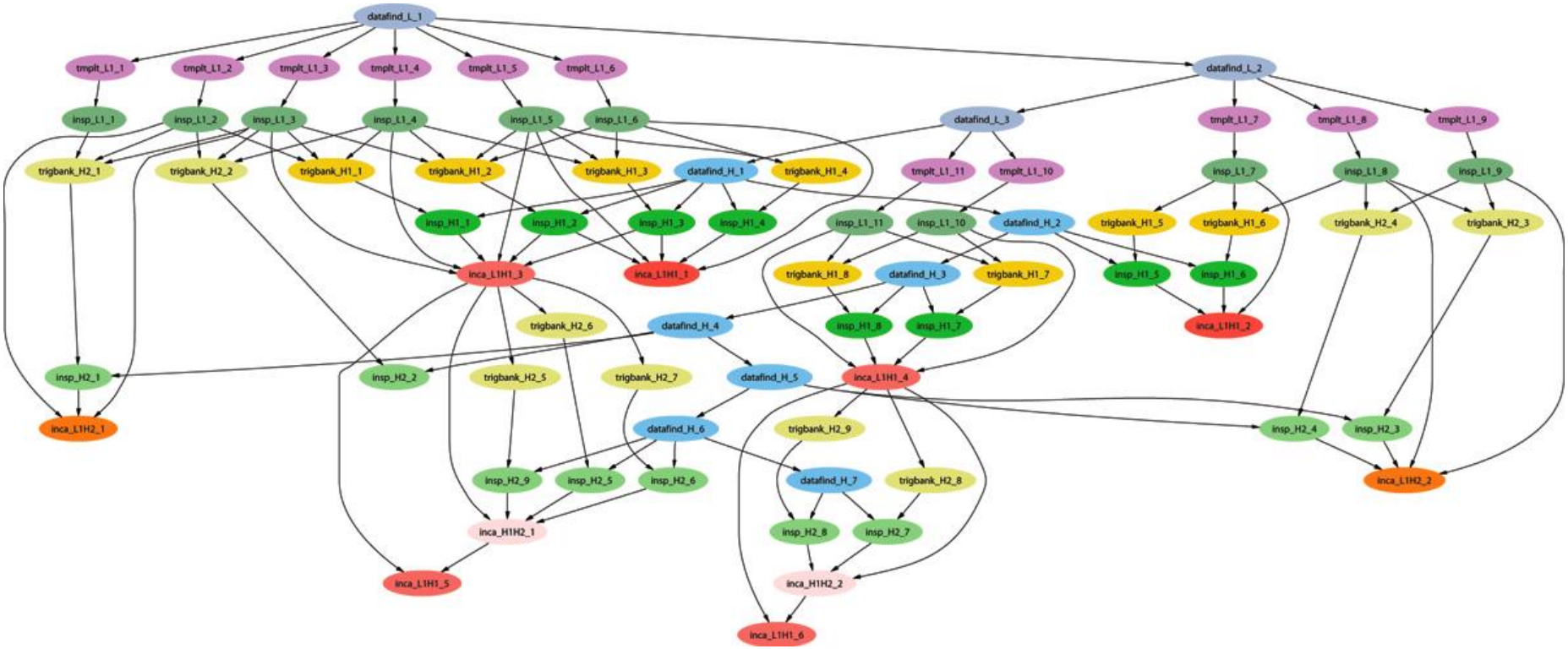
OSG helps LIGO confirm Einstein's theory

Einstein predicted gravitational waves over 100 years ago. Open Science Grid (OSG) resources are helping the NSF-funded Laser Interferometer Gravitational-Wave Observatory (LIGO) prove he was right.

> ***Thus far, LIGO has consumed almost four million hours on OSG — 628,602 hours were on Comet and 430,960 on Stampede resources. OSG's Brian Bockelman of the University of Nebraska-Lincoln and Edgar Fajardo from the SDSC used HTCondor software to help LIGO implement their Pegasus workflow on 16 clusters at universities and national labs across the US.***

- “When a workflow might consist of 600,000 jobs, we don’t want to rerun them if we make a mistake. So we use DAGMan (Directed Acyclic Graph Manager, a meta-scheduler for HTCondor) and Pegasus workflow manager to optimize changes,” added Couvares. “The combination of Pegasus, Condor, and OSG work great together.” Keeping track of what has run and how the workflow progresses, Pegasus translates the abstract layer of what needs to be done into actual jobs for Condor, which then puts them out on OSG.

Example of a LIGO Inspiral DAG (Workflow)



HTC is about sharing across many **jobs**, many **users**, many **servers**, many **sites** and (potentially) long running **workflows**

A job submitted to a batch service consists of an Acquisition Request (**AquR**) and a Job Description (**JobD**).

The Provision Manager (**Pman**) of the service provisions the resources and then runs the job on theses resources via the Job Launcher (**JaL**)

Most batch services manage a static collection of resources

HTCondor uses a matchmaking process to dynamically **acquire** resources.

HTCondor uses a matchmaking process to **provision** them to queued jobs.

HTCondor launches jobs via a task delegation protocol.

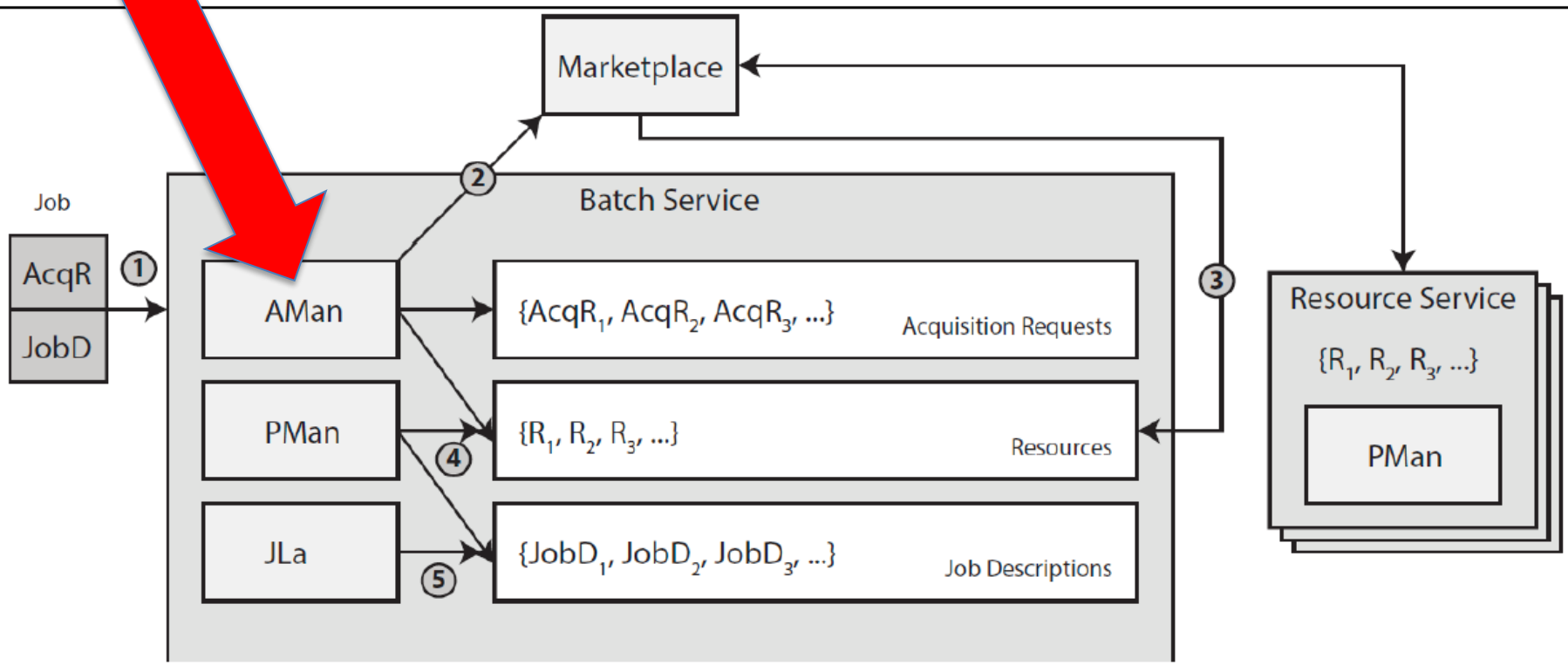
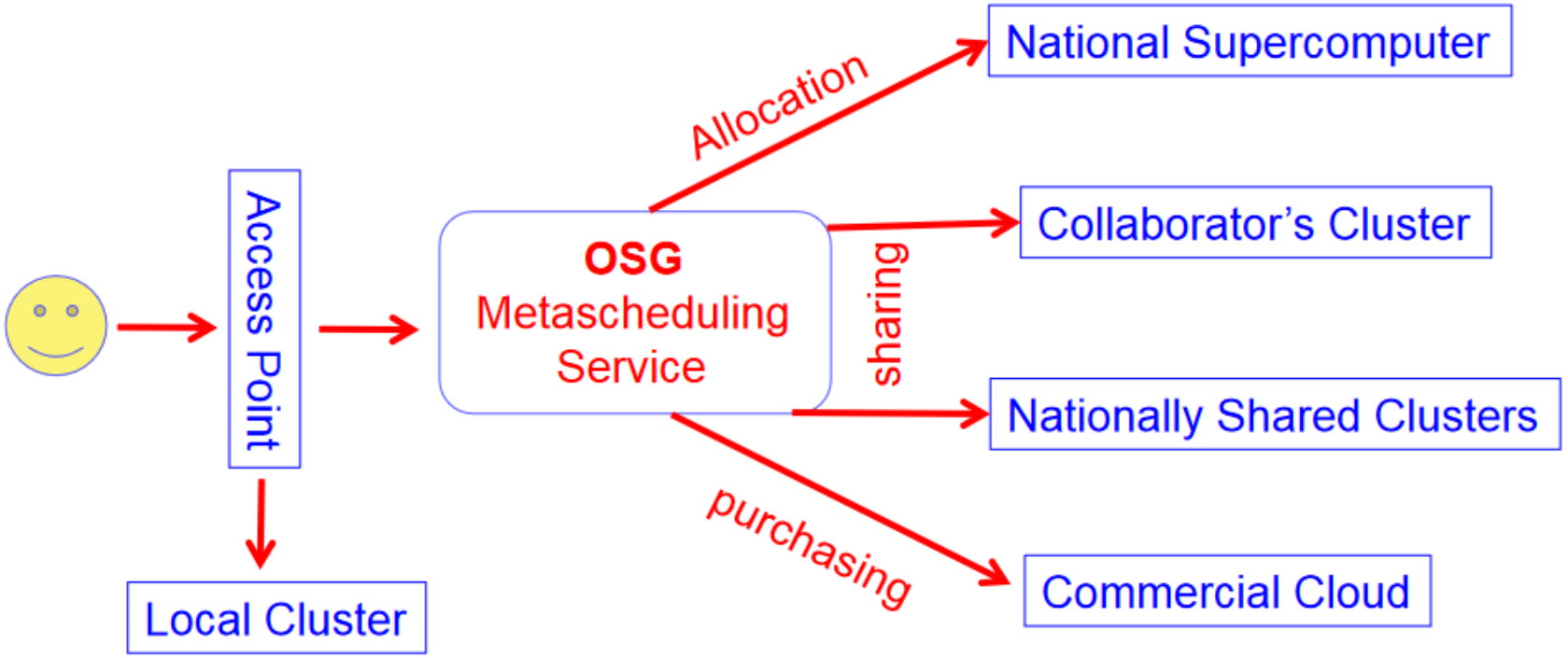


Figure 1: The HTC Framework: (1) A user submits job, composed of an AcqR and a JobD. (2) The AMan requests a resource compatible with an AcqR. (3) The MarketPlace offers a compatible Resource to the batch service. (4) The Pman selects a job description and Resource to send to (5) the JLa, which runs the job.

The OSG Vision



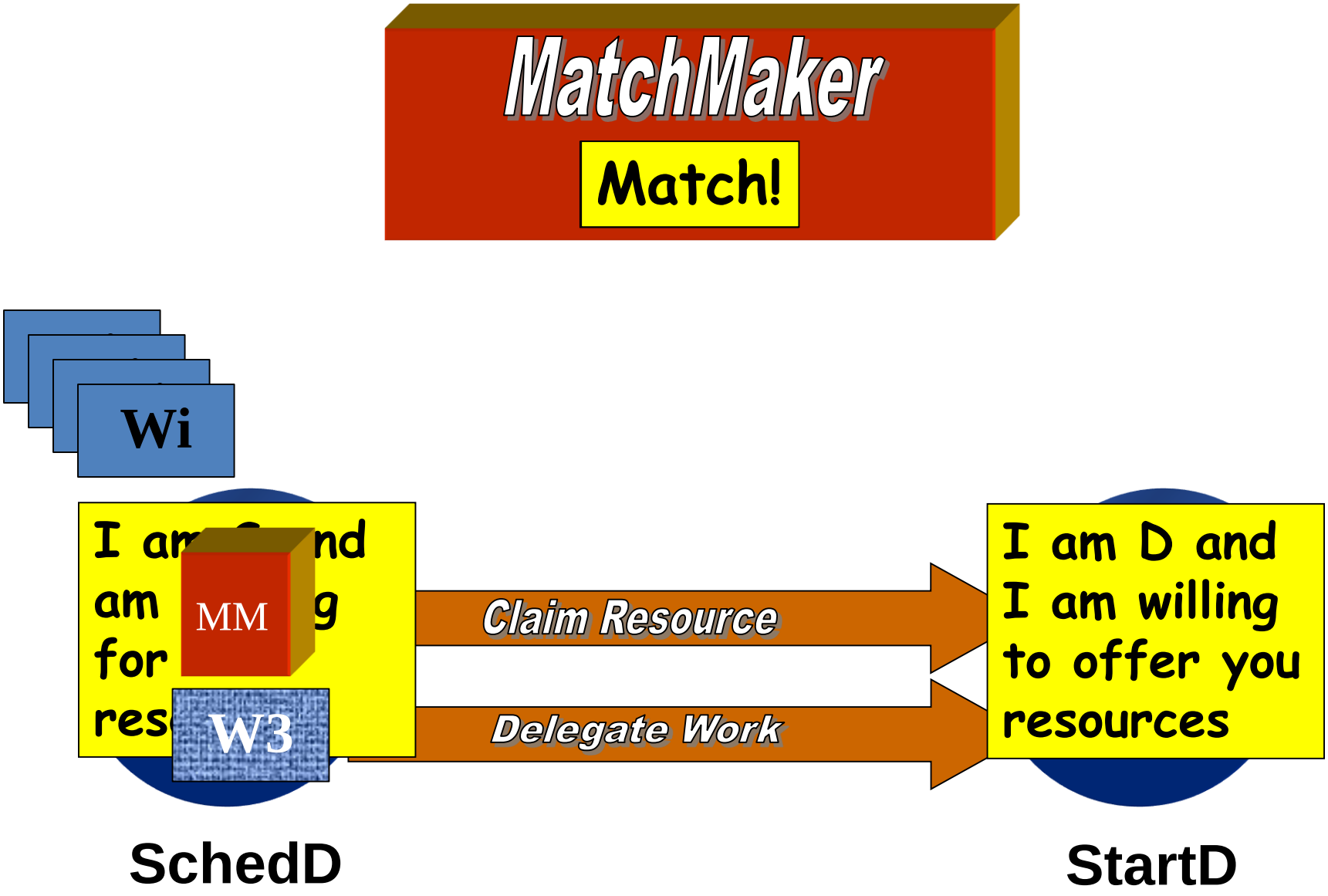
OSG integrates computing across different resource types and business models to allow Campus IT to offer a maximally flexible HTC environment to your researchers.

Traditional (low frequency) Capacity Planning

Turning $s into computing power

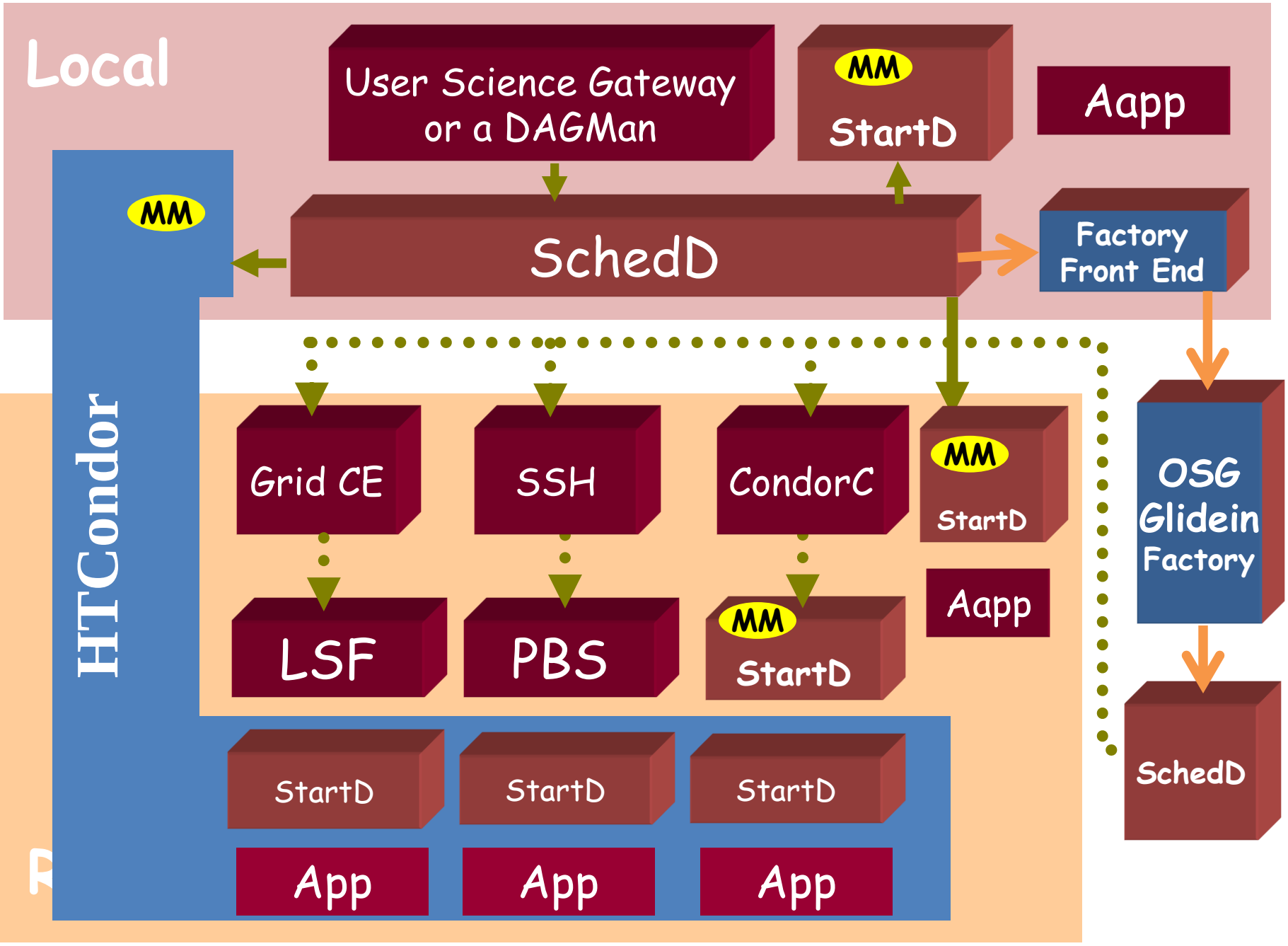
- Collect workload characteristics and customer (performance) metrics
- Understand the cost-performance profile of the hardware and software options
- **Acquire** (select, purchase, install) the resources and place them under the control of a **batch service**
- Live with your decision for (5-8) years

Here is what the OSG offers today with the support of HTCondor technologies


MatchMaker
Match!
Wi
I ar nd
am g
for
res
MM
W3
Claim Resource
Delegate Work
I am D and
I am willing
to offer you
resources
SchedD
StartD

HTCondor 101

- Jobs are submitted to the HTCondor **SchedD**
- A job can be a **Container** or a **VM**
- The **SchedD** can **Flock** to additional **Matchmakers**
- The **SchedD** can delegate a job for execution to a **HTCondor StartD**
- The **SchedD** can delegate a job for execution to a another **Batch system**.
- The **SchedD** can delegate a job for execution to a **Grid Compute Element (CE)**
- The **SchedD** can delegate a job for execution to a **Commercial Cloud**


Local
User Science Gateway or a DAGMan
MM
StartD
Aapp
MM
SchedD
Factory Front End
HTCondor
Grid CE
SSH
CondorC
MM
StartD
OSG Glidein Factory
LSF
PBS
MM
StartD
Aapp
StartD
StartD
StartD
SchedD
App
App
App

Researcher or VOs may have …

- **Resources** they own and therefore fully control
- An **allocation** of resources on shared campus/national computing facility
- **"Fair Share" privileges** on shared campus/national computing facilities
- **Opportunistic Resources** provided by collaborators
- **Funding** to purchase resources from a commercial cloud provider

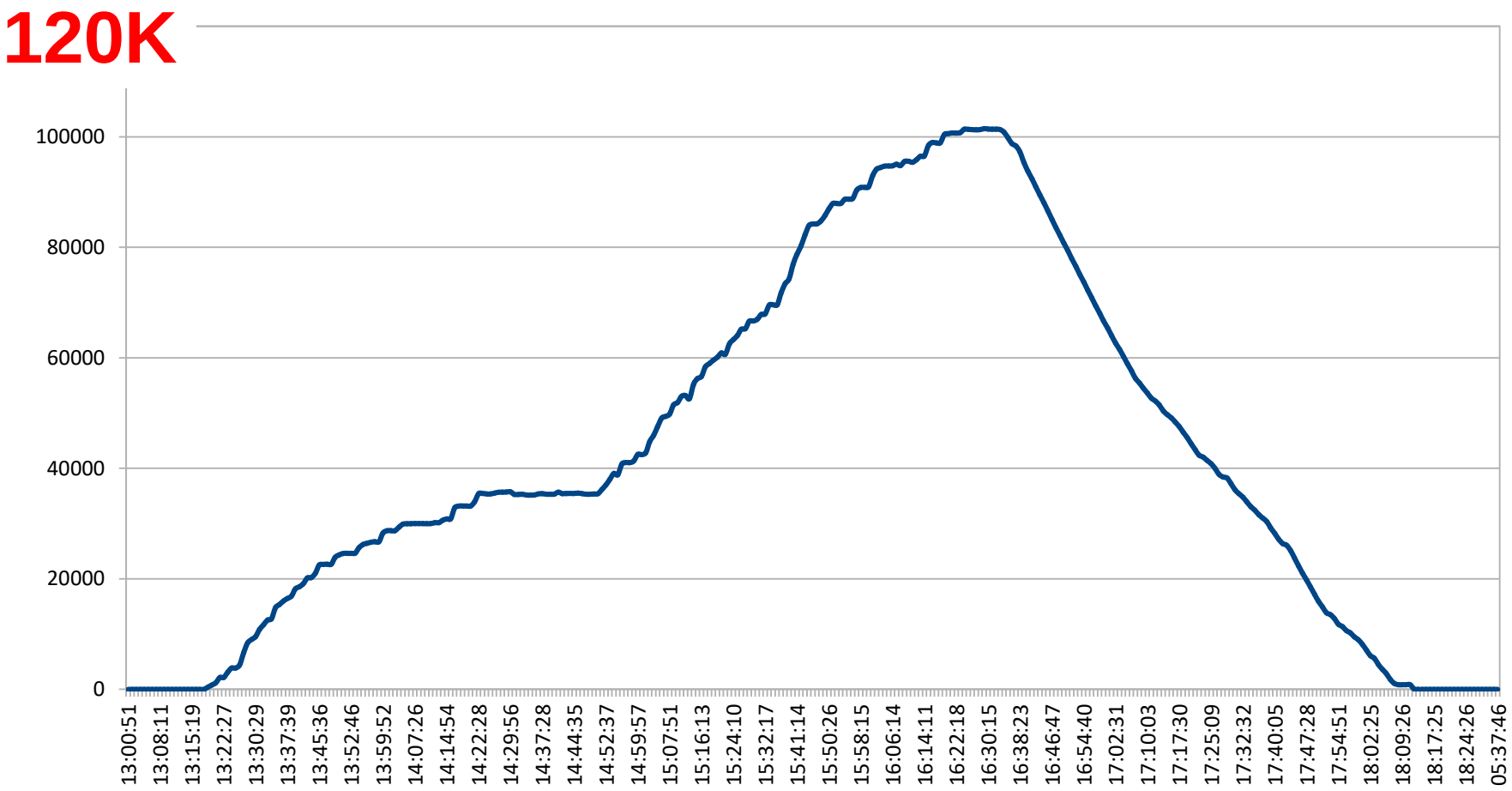
Commercial clouds offer to individuals with money …

- **Unbounded** on demand capacity for (almost) as long as needed
- A **variety** of cost/performance option for processing and storage resources
- **Dynamic cost** structures that track demand and supply
- **Diverse** (and competing) suppliers of computing resources and associated services

Next generation (High frequency) Capacity Planning when resources can be rented by the minute or by the hour

Here is what we can do today with the **Condor-Annex** Utility

What $1.5K can do for you



13:00 **Jobs running on AWS Spot instances** **18:00**

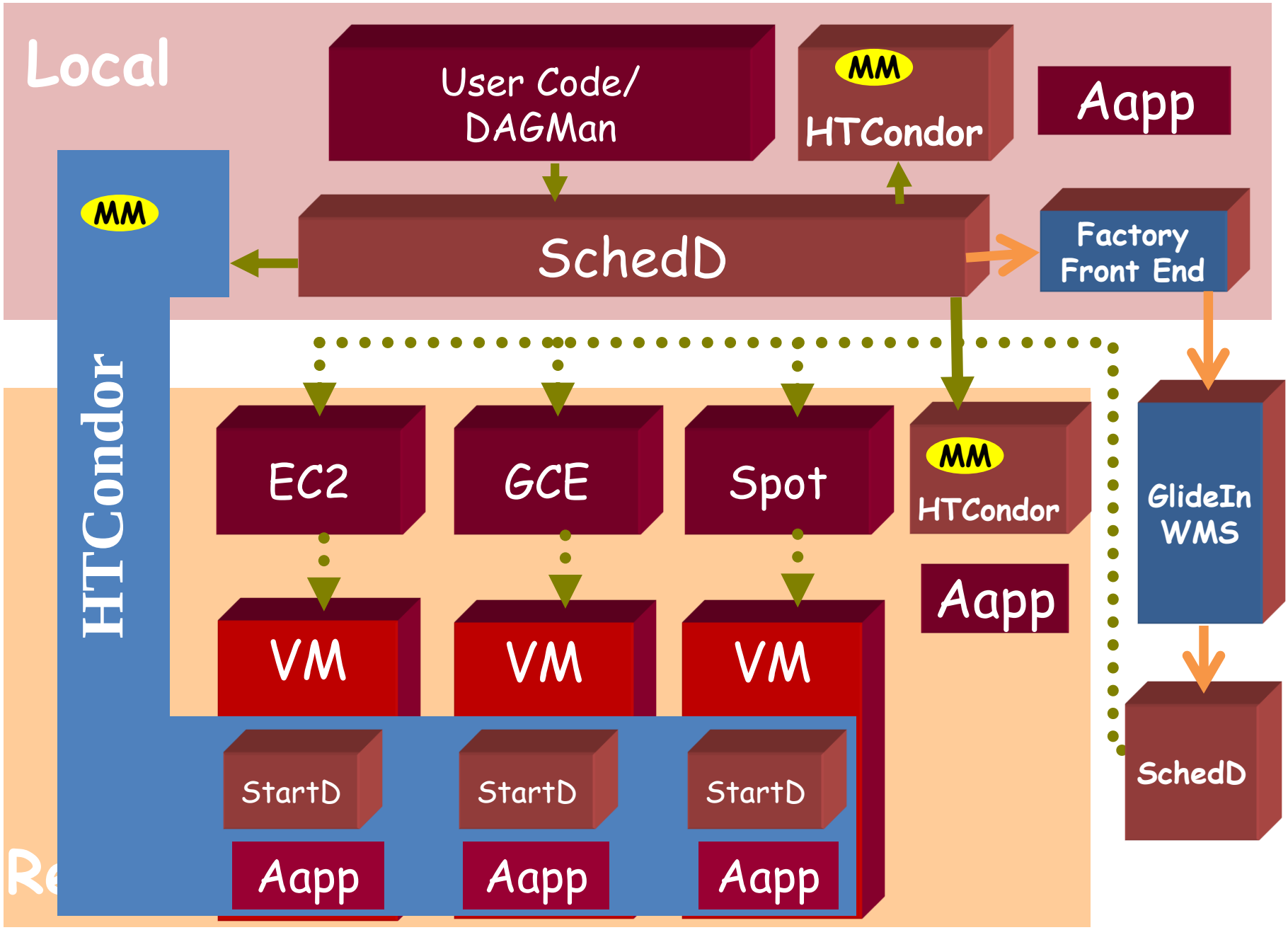
Annex means Addition

- An annex is “a building joined to main building, providing additional space or accommodations.”
  - extra cores
  - GPUs or larger main memories
  - specialized policies
- To `condor_annex` is “to append or add as an extra or subordinate pool.”

Annex Lifecycle

1. User requests resources (number, duration).
2. Then `condor_annex` starts instances.
3. Instances join pool.
4. Instances stop spending your money:
   - if they become idle, or
   - after the duration.

Here is what we can do today with the **GlideIn Workflow Management System (WMS)**


Local
User Code/
DAGMan
MM
HTCondor
Aapp
MM
SchedD
Factory
Front End
HTCondor
EC2
GCE
Spot
MM
HTCondor
Aapp
GlideIn
WMS
VM
VM
VM
StartD
StartD
StartD
Aapp
Aapp
Aapp
SchedD

SC16 CMS Demonstrator

Target: generate 1 Billion events in 48 hours during Supercomputing 2016 on Google Cloud via HEPCloud

35% filter efficiency = stage out 380 million events → 150 TB output

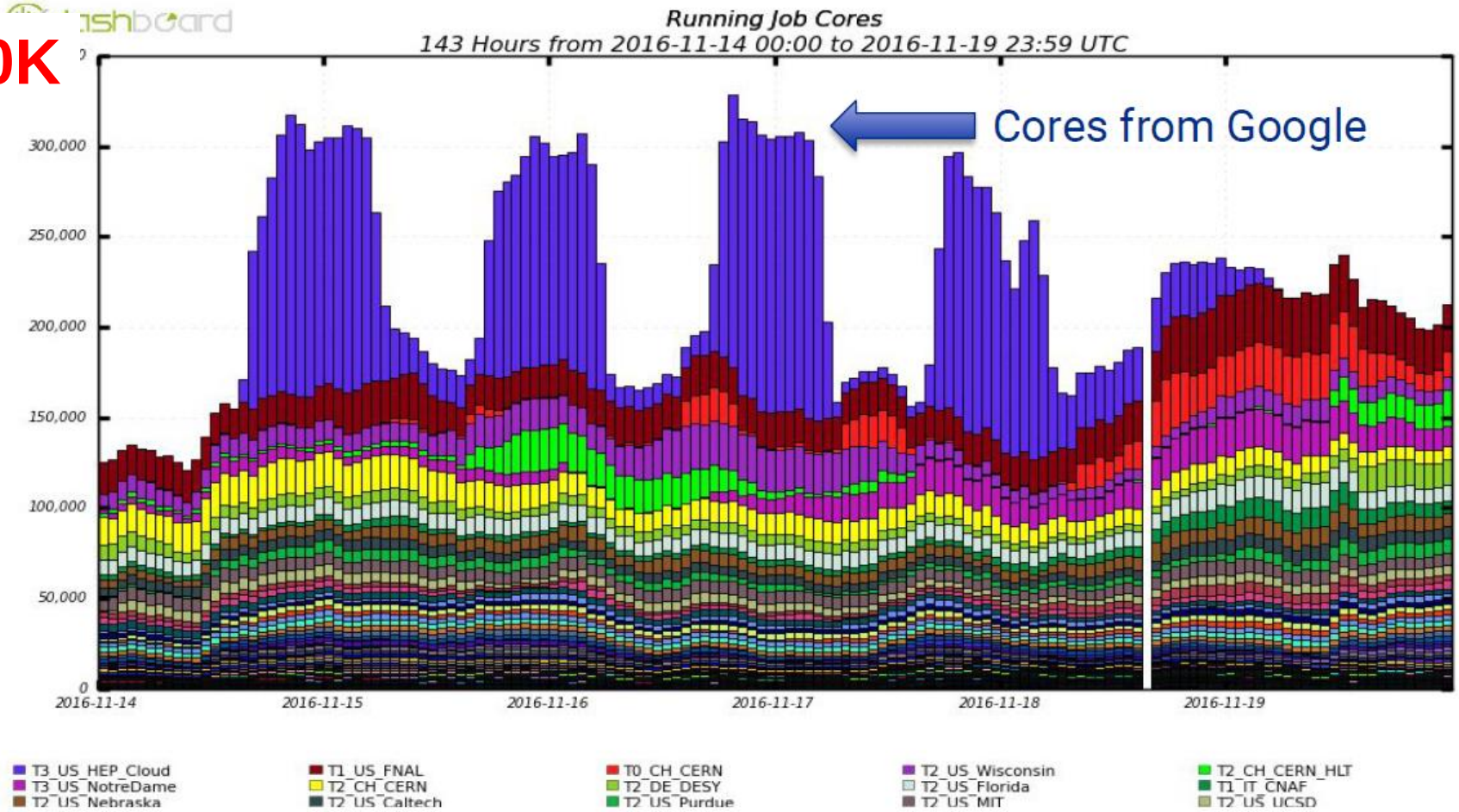
Double the size of global CMS computing resources

CMS @ Google – preliminary numbers

- 6.35 M wallhours used; 5.42 M wallhours for completed jobs.
  - 730172 simulation jobs submitted; only 47 did not complete through the CMS and HEPCloud fault-tolerant infrastructures
  - Most wasted hours during ramp-up as we found and eliminated issues; goodput was at 94% during the last 3 days.
- Used ~$100k worth of credits on Google Cloud during Supercomputing 2016
  - $71k virtual machine costs
  - $8.6k network egress
  - $8.5k disk attached to VMs
  - $3.5k cloud storage for input data
- 205 M physics events generated, yielding 81.8 TB of data

All Managed by HTCondor!

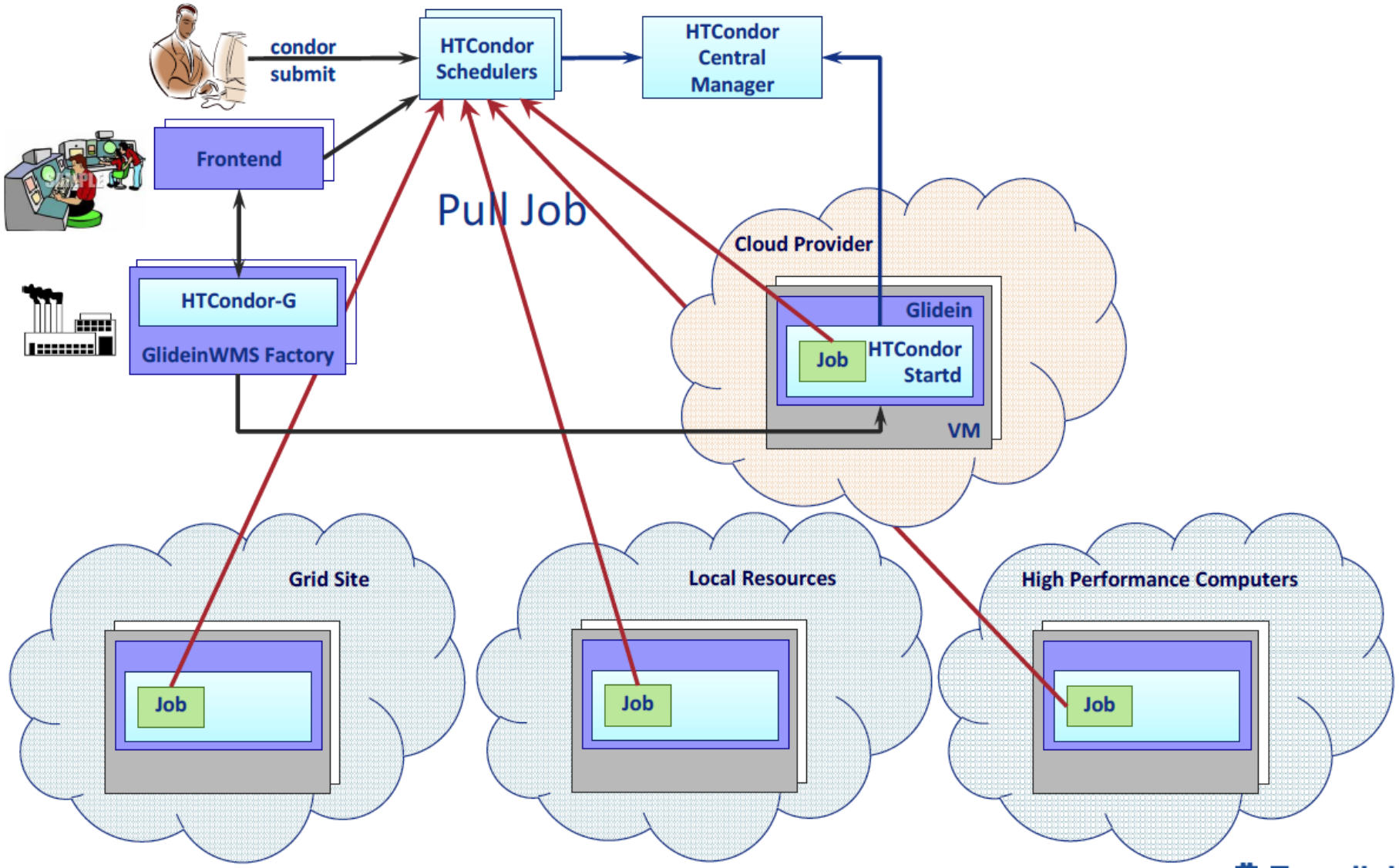
350K



Google Cloud

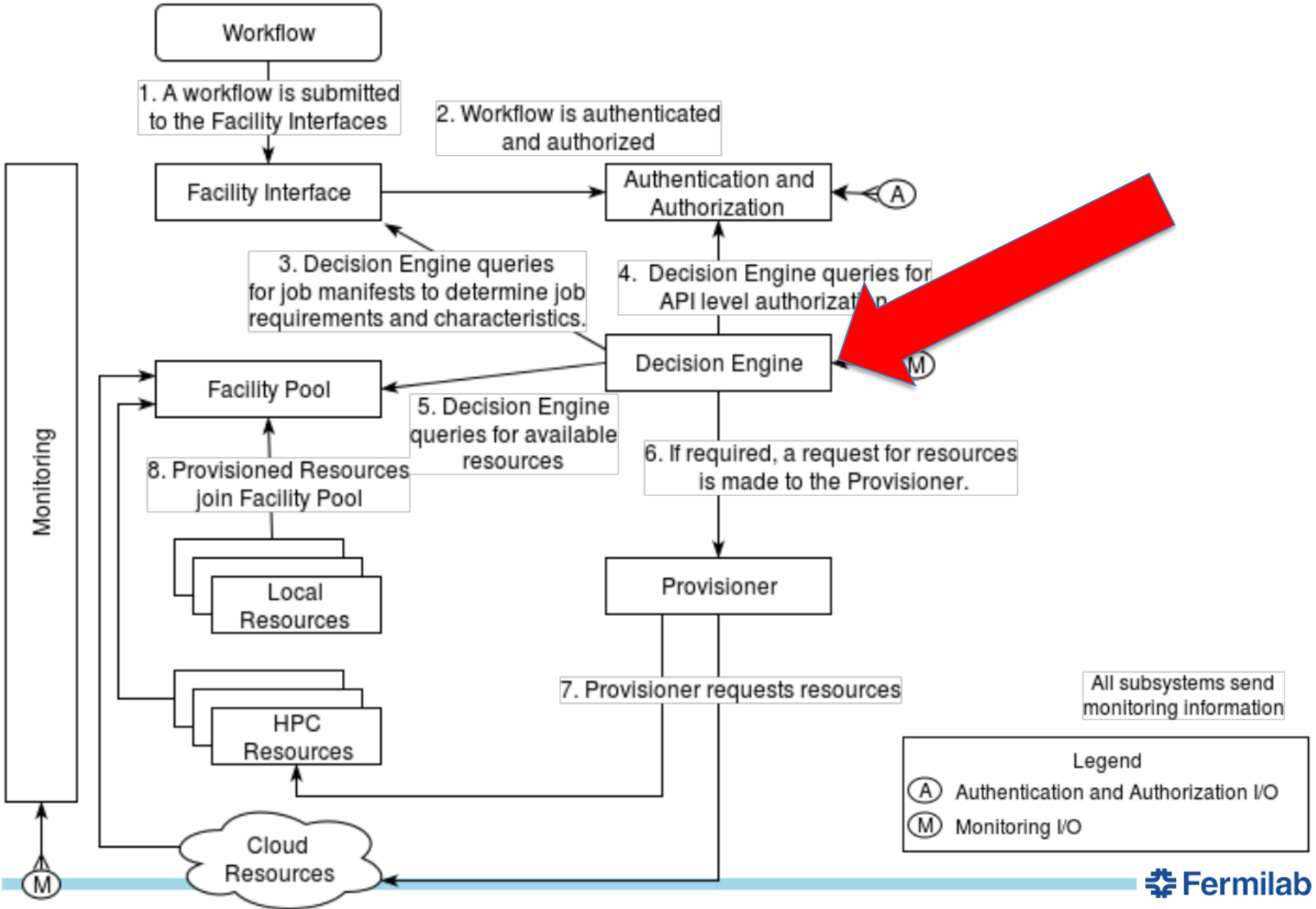
HEPCloud is an R&D project led by the Fermi computing division

HEPCloud – glideinWMS and HTCondor



Fermilab

HEPCloud Architecture

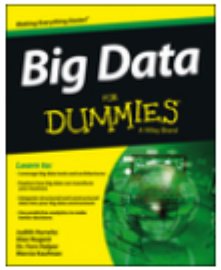


Decision Engine will have to implement **on-the-fly capacity planning** to control acquisition and release of resources

Many Challenges Ahead

- Language to define policies
- Software that manages (real) money
- Validation and verification of policies and software tools
- Risk management
- Accounting and Auditing
- Integration with DAGMan (workflows)

DISTRIBUTED COMPUTING BASICS FOR BIG DATA



RELATED BOOK

Big Data For Dummies

By Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman

If your company is considering a big data project, it's important that you understand some distributed computing basics first. There isn't a single distributed computing model because computing resources can be distributed in many ways.

Timeline of projects