Principles for High Throughput Computing

Alain Roy <roy@cs.wisc.edu>
Open Science Grid Software Coordinator

Open Science Grid

Reminder: What is HTC?

- An approach to distributed computing that focuses on long-term throughput, not instantaneous computing power
  - We don't care about operations per second
  - We care about operations per year
- Implications:
  - Focus on reliability
  - Use all available resources
  - That slow four-year old cluster down the hall? Use it!

Q: What system am I describing?

- If you adopt system **X**, you will get:
  - High availability
  - High reliability
  - Resource sharing
  - Automatic load sharing
  - Easy expansion in both capacity and function
  - Good response to temporary overloads

Condor? OSG?

Amazon EC2? Other Clouds?

Open Science Grid

Actually…

- Those as promises for distributed data processing systems from the 1970s, more than 30 years ago!
  - Drawn from paper by Enslow, 1978
- Yet they sound very similar to the promises for today's distributed systems, grids, and clouds

Miron Livny

- Founder and leader of the Condor Project since mid-80's
- PI and technical director of the Open Science Grid
- Coined the term "high throughput computing"
- Has a *principled* approach to high throughput computing research

Miron likes to remind us:

The words of Koheleth son of David,
king in Jerusalem ~ 200 B.C.

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)

There is nothing new

Distributed data processing	Virtualization
Cluster computing	Peer-to-peer
Grid computing	Client-server
Cloud computing	Cyberinfrastructure…

New terms, new hype…

But the underlying problems are the *same*.

The principles to address the problems are the *same*.

Really? Aren't clouds new and cool?

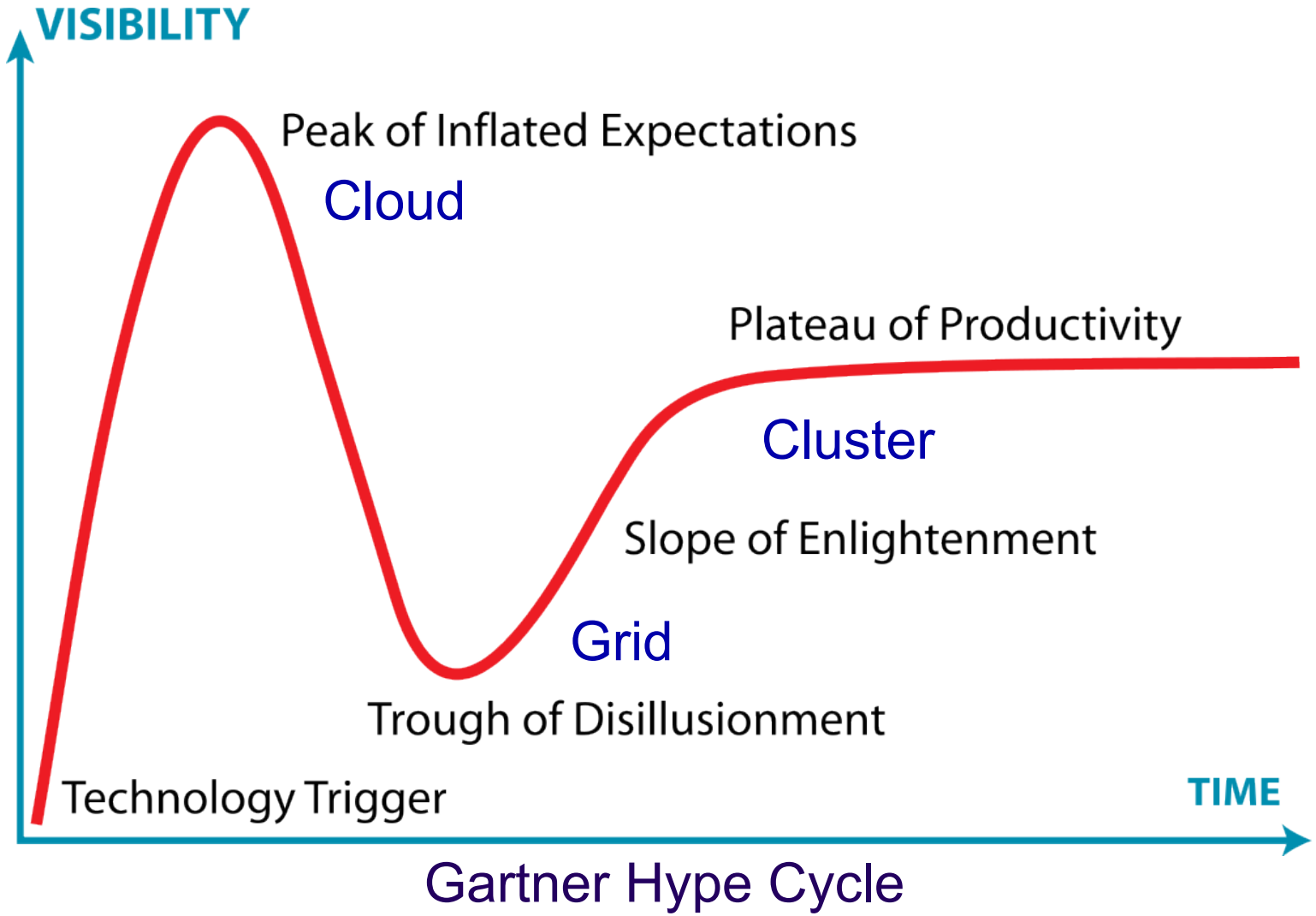
- Amazon EC2 makes it easy to create virtual machines on the fly. That's new, right?
  - Condor matchmaking:
    Allocate a computer + Start a job
  - Amazon EC2:
    Allocate a virtual machine + Start it
  - Conceptually a virtual machine is just another kind of job: you start it, you stop it, you provide policies to control it

Open Science Grid

Clouds are a little bit new

- In OSG and XSEDE, it is free to allocate a computer
  - OSG: Allocate as many as you like (when they're available)
  - XSEDE: You get a free allocation if your science is "good"… And you can request a free extension
- In Amazon EC2: you need a credit card
  - No more free lunches
  - Sure you can have more time—if you pay
  - You can evaluate "build local cluster" vs "outsource to Amazon"

Don't fall for the hype



http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp

Open Science Grid

Instead:

- Know the principles
- Understand how the principles apply

- And you will be able to use the next technology in the next hype cycle to help your science

- So let's look at some principles

First, let's define distributed computing

"You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done."

–Leslie Lamport

First, let's define distributed computing

- Distributed computing systems:
  1. Have a **multiplicity** of resources
  2. Are **physically distributed**
  3. Provide **unity of system operation**
  4. Have **system transparency**
  5. Have **cooperative autonomy**

From *What is a "Distributed" Data Processing System?*, by Philip Enslow, IEEE Computer, Vol 11, No. 1, January **1978**, p 13-21.

Open Science Grid

1. Multiplicity of resources

- There should be multiple general-purpose resource components—physical and logical—that can be assigned to tasks
- Homogeneity is not essential
- More resources helps maintain reliability and performance

2. Physical distribution

- Components should be physically distributed
- They should be connected via network with a two-party cooperative protocol
  - Protocol allows requests to be refused based on local knowledge
  - Example: HTTP

3. Unity of System Operation

- All components in the system should be unified to achieve a common goal
- Unification can be via
  - Shared software, or
  - Well-defined policies
- There are multiple centers of control
  - No critical paths
  - No critical components

Open Science Grid

4. System Transparency

- It feels like a “single virtual machine”
- When requesting a service, the user does not need to know:
  - Which physical component will do it
  - Where that component is located

Open Science Grid

5. Cooperative autonomy

- There is cooperation: common policies and goals
- Resources make their own decisions:
  - Accept or refuse requests
  - Based on local policies for use of resource

Does your laptop fit?

Multiplicity?	Yes, multiple CPUs/cores
Physically distributed?	No, all in one box
Unity of system operation?	Yes, one OS
System transparency?	Yes
Cooperative autonomy?	No, centrally controlled

Answer: your laptop alone does not do distributed computing

Are glideins distributed computing?

Multiplicity?	Yes, multiple CPUs/cores
Physically distributed?	Yes, across the world
Unity of system operation?	Yes, one system & one goal
System transparency?	Yes
Cooperative autonomy?	Yes, sites are autonomous

Answer: glideins are a distributed computing system

On your own: A single web-server? A load-balanced set of web servers with db backend?

Four principles for distributed computing

These principles are guiding principles for the Open Science Grid

- Copied from the recent OSG Proposal

Four principles for distributed computing

1. Resource Diversity

The system must be flexible enough to accept many types of resources, software and services

2. Dependability

Throughput must be tolerant to faults: there will always be services or resources that are not available

Four principles for distributed computing

3. Autonomy

You must allow resource providers from different organizations to share
You must allow them to preserve local autonomy to set policies and select technologies

4. Mutual Trust

You must support complex trust relationships across organizations and software tools

Do we apply these principles for glideins in OSG?

Let's check them out…

1. Glideins & resource diversity

- Glideins can submit to different kinds of compute elements:
  - GRAM, CREAM, NorduGrid
- Compute elements work with different clusters
  - Condor, Torque, PBS, LSF, SGE
- Glidein software supports multiple OSes and architectures:
  - 32 & 64 bit
  - Scientific Linux 5 & 6

2. Glideins and dependibility

- GlideinWMS uses Condor:
  - Reliable queue to manage jobs
  - Tolerance to resources appearing and disappearing
- Glideins use multiple sites:
  - Individual sites come and go

Open Science Grid

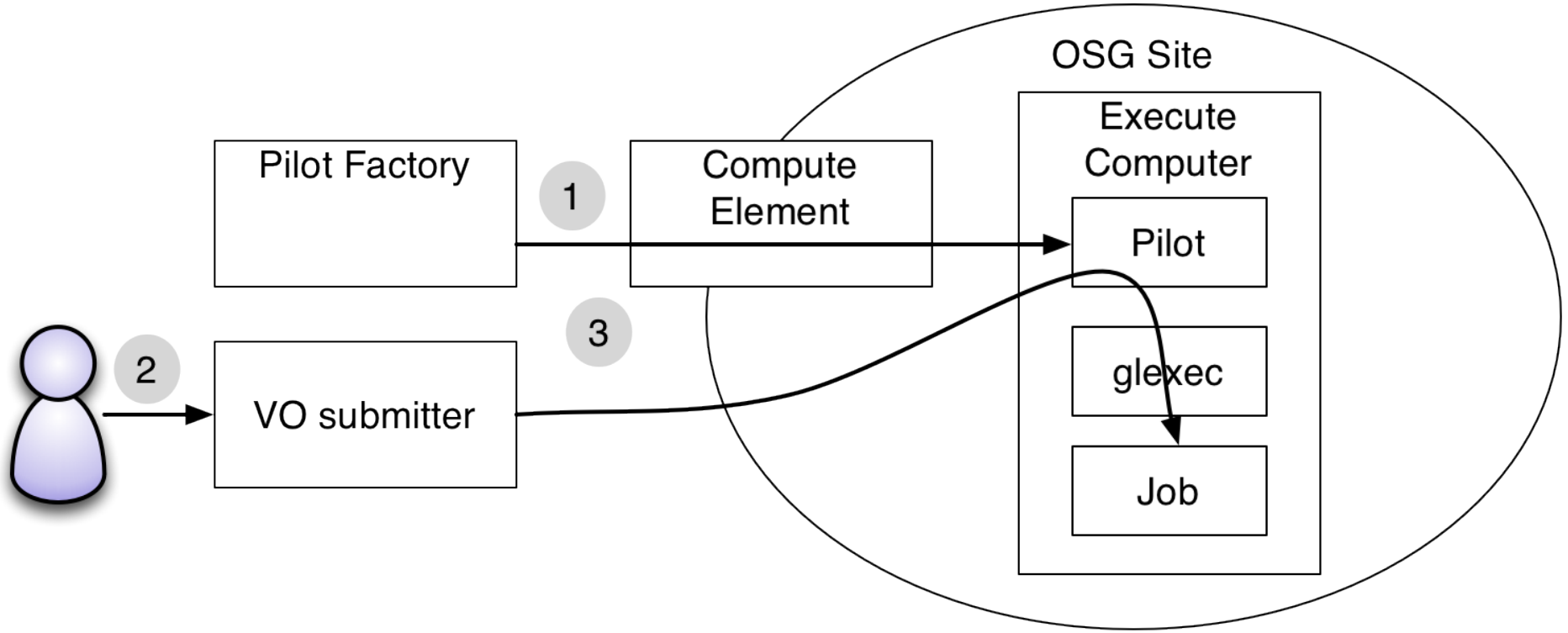
3. Glideins and Autonomy

- Sites are autonomous
  - They choose to accept VOs and users
  - They can limit how many pilot jobs run
  - They can limit how long pilot jobs run
  - They can prefer local users (preempt glideins)
- Glideins are autonomous
  - Factory decides how many pilots to submit
  - Factory controls policy on running pilots
  - Glidein pool has policies for how glidein users share resources

4. Glideins and mutual trust

- Glidein pilot jobs are authenticated and authorized
  - Based on X.509 certificates
- Glexec
  - Some sites are uncomfortable that pilot users is not the owner of the job
  - Glexec is voluntary way to authorize and/or log the owner of the jobs

Glexec


OSG Site
Execute Computer
Pilot Factory
1
Compute Element
Pilot
3
2
VO submitter
glexec
Job

How should you apply the principles?

- Resource diversity
  - Can your application run on multiple OSes?
  - In multiple environments?
  - If you gain access to more computers, how hard will it be for you to use them?
- Dependability
  - Use a system that recovers from faults
  - Write your code to expect failures and recover (or report good errors!)
    c.f. Greg's talk from this morning

How should you apply the principles?

- Autonomy
  - Expect resources to be autonomous: they make choices for their own benefits
  - Resources come and go unpredictably

How should you apply the principles?

- When someone sells you a new technology, or software or …
  Ask:
  - Can it handle diverse resources?
  - Is it dependable?
  - Are components autonomous?
  - Does it manage trust relationships?
  - Does it improve on the ability to provide these fundamental principles, or is it merely a shiny new version of what we already have?

These apply at all scales

- The principles apply:
  - For a single computer
  - For a cluster
  - For a campus
  - For a grid/cloud (like OSG)

Open Science Grid

Questions?

- Questions? Comments?
  - Feel free to ask me questions later:

    Alain Roy <roy@cs.wisc.edu>
- Upcoming sessions
  - Now – 2:45
    - Break
  - 2:45 – 4:20
    - High Throughput Computing Showcase