Project Sigrid: The Simplest Possible Grid Computing Platform

Tim Bray

Director of Web Technologies
Sun Microsystems
http://www.tbray.org/ongoing/

TS-3108

Goal of This Talk

Learn what Grids are good at and not good at, what kinds of Grids there are, and about Project Sigrid, a simple Web-style grid framework

Agenda

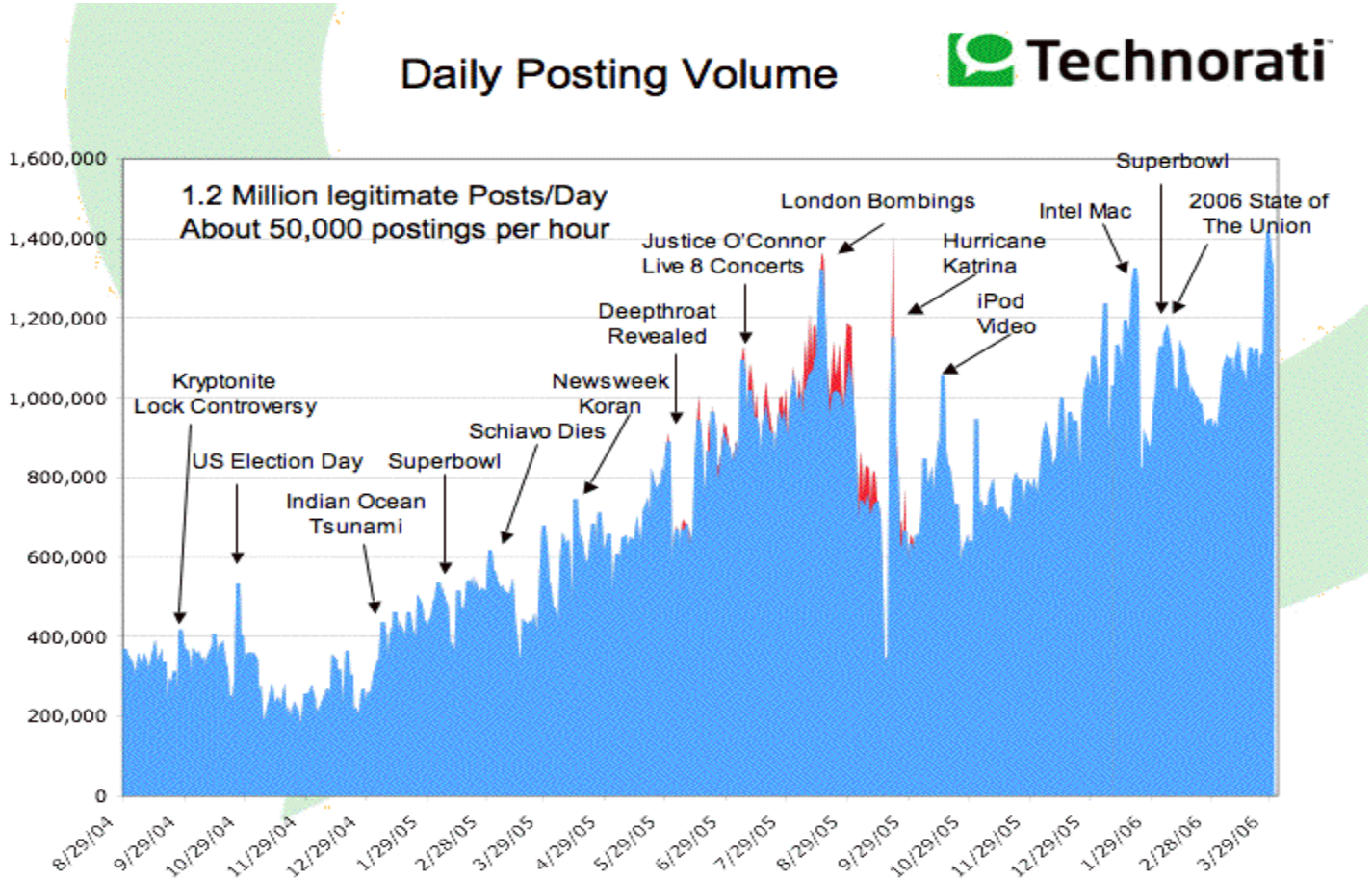
Why Grids? Why Not?

Grid History:
Batch and Service Orientation

Existing Grid Frameworks (How to Say "Hello World" 10^8 Times in Parallel)

Project Sigrid

Why Grids? Big Problems!



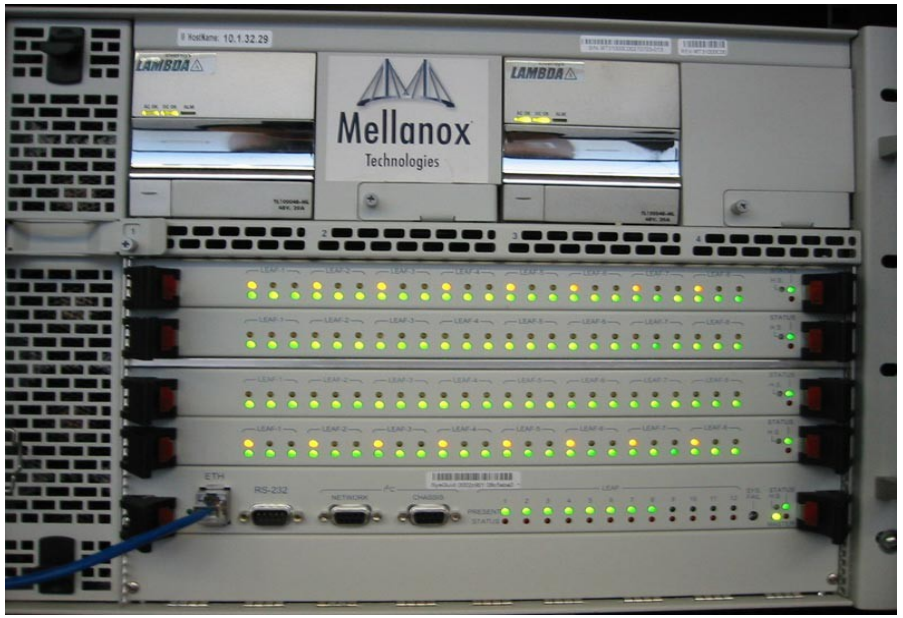
Source: Dave Sifry, Technorati (www.sifry.com/alerts)

Big Problem: One Year of Blog Search

- 1.2 million posts/day; average ~100 words/post
- Per year: ~44 billion words, ~220G text, ~500G full-text index
- About 13 new articles per second, so ~1,000 index updates/second
- One million searches/day, ~11.5/sec
- No single computer can handle the update/query load
- You really don't want to involve disks

Some Performance Facts

- Memory is a lot faster than disks: a big 4-way server running SolarisTM ZFS, highly parallelized, can do maybe 500 random seeks/second
- It's faster even if you have to go across a data center network to another computer's memory (Infiniband, 10G ethernet)



Source (image): Virginia Tech (tcf.vt.edu)

Slogan

Memory is the new disk
Disk is the new tape

Blog Search: Grid Solution

- T2000 with 32G RAM: $27K
- 20xT2000 = 640G: ~$500K list
- Should be able to handle updates and quite a few million queries/day

Source: Sun.com

Why Not Put Everything on the Grid?

In 2003, there was rough price parity between:

- One database access
- Ten bytes of network traffic
- 100,000 instructions
- 10 bytes of disk storage, and
- 1 megabyte of disk bandwidth

Source: Jim Gray, Microsoft Research (research.microsoft.com/research/pubs/view.aspx?tr_id=655)

Why Not Put Everything on the Grid?

In other words:

- CPU is affordable
- Memory is cheap
- Disk is free
- Moving data is expensive!

Grid Economics

- Maximize the ratio of computation to data traffic
  Example: SETI@Home, render farms
- Put the data near the computation
  Example: Google GFS + MapReduce

Agenda

Why Grids? Why Not?

Grid History:
Batch and Service Orientation

Existing Grid Frameworks (How to Say "Hello World" 10^8 Times in Parallel)

Project Sigrid

What Do Grids Do Today?

- Render movies
- Simulate silicon
- Price derivatives
- Model oil fields
- Build Web search indices
- Search the Web

What Do Grids Do Today?

- Render movies
- Simulate silicon
- Price derivatives
- Model oil fields
- Build Web search indices
- Search the Web

} Batch

A Service-Oriented Grid

For example, Google

- Always online
- Applications never stop running
- The data lives in the grid

Agenda

Why Grids? Why Not?

Grid History:
Batch and Service Orientation

Existing Grid Frameworks (How to Say "Hello World" 10^8 Times in Parallel)

Project Sigrid

The Grid Landscape

	Real	Speculative
Batch	MPI, MapReduce, SGE, DRMAA	
Service-Oriented	Google, Yahoo, etc.	OGSA, Rio, Gridbus

Batch API: MPI (and MPI-2)

MPI 1994, MPI-2 1997

- FORTRAN and C APIs for message-passing and parallel I/O; no Java™ technology yet
- Lots of message-distribution patterns: point-to-point, broadcast, etc.
- Synchronous and asynchronous
- Language-independent (sort of) data binding
- “Reduce” operation

Source: MPI Forum (mpi-forum.org)

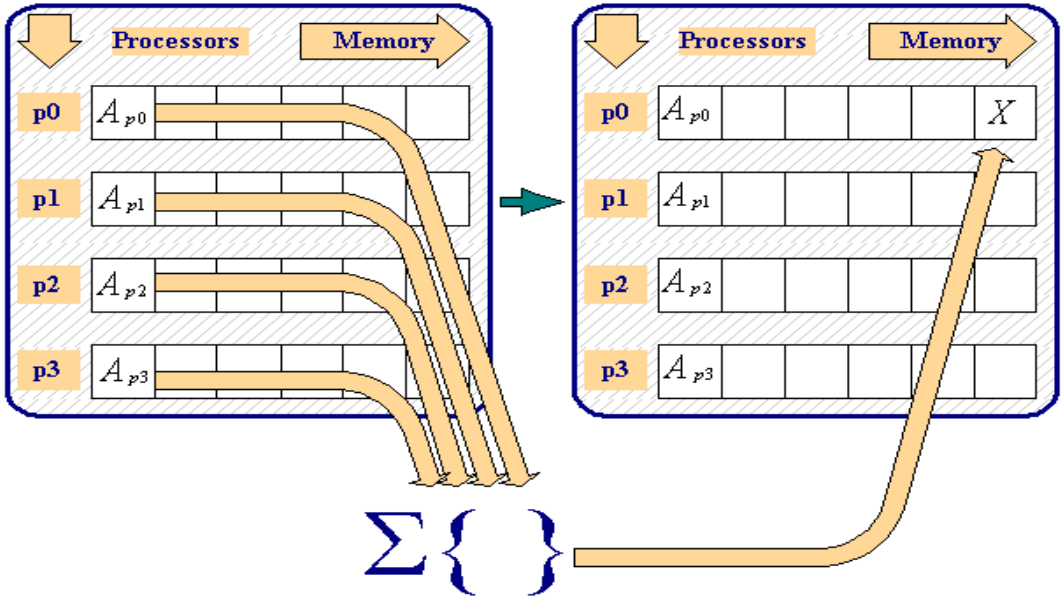
MPI Features

- "Processors" abstract across arbitrary number of nodes
- Broadcast like this:

```
int MPI_Bcast ( void* buffer, int count, MPI_Datatype
datatype, int rank, MPI_Comm comm );
```

- Can divide processors into matrices and sub-matrices for dividing up work

MPI's “Reduce” Function




```
count = 1;
rank = 0;
MPI_Reduce( &a, &x, count, MPI_REAL,
            MPI_SUM, rank, MPI_COMM_WORLD );
```

MapReduce

A central piece of Google infrastructure

- A “map” function:
  `map (in_key, in_value) → list(out_key, intermediate_value)`
  - Processes input key/value pair
  - Produces set of intermediate pairs
- A “reduce” function
  `reduce (out_key, list(intermediate_value)) → list(out_value)`
  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

Source: Google (labs.google.com/papers/mapreduce.html)

MapReduce Example: Word Counter

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
      EmitIntermediate(w, "1");


reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
      result += ParseInt(v);
    Emit(AsString(result));
```

MapReduce Implementation

- C++ ,runs across a cluster of a few thousand machines
- Assumes use of Google Filesystem to get data close to computation
- Can tolerate machine failures and even bugs (by detecting repeated failures on the same data)
- Redundant execution on CPUs that free up first
- Test: Scan 1010 100-byte records to extract records matching a rare pattern (92K matching records): 80 sec.
- Test: Sort 1010 100-byte records: 839 sec.

MapReduce Is Coming to the Java Platform!

- Doug Cutting, of Lucene and Nutch fame
- See http://svn.apache.org/repos/asf/lucene/nutch/
- Single-CPU only so far

Source: Tom White (weblogs.java.net/blog/tomwhite/archive/2005/09/mapreduce.html)

The Sun Grid Engine by Example

- Step1 initializes, reads a data file “input.txt”, writes three intermediate files
- Step2 processes them in parallel, no dependencies
- Step3 finalizes

Source (all SGE data): Network.com

Sun Grid Engine: Three-way "Hello World"

```
>cat run.sh:
#!/bin/sh
qsub -N step1                    -b n step1.sh
qsub -N step2 -hold_jid step1 -b n step2.sh
qsub -N step2 -hold_jid step1 -b n step2.sh
qsub -N step2 -hold_jid step1 -b n step2.sh
qsub          -hold_jid step2 -b n step3.sh
>zip app.zip run.sh input.txt step*.sh
... zip output elided ...

... upload app.zip and run via Web GUI ...

... fetch output via Web GUI ...
```

Sun Grid Engine Environment

- You're on your own subnet; no Internet connection
- You have a locally-rooted filesystem
- There is demand for service-oriented capabilities, but there are complexity, security, and safety issues

Global Grid Forum

- First met in 2001
- Currently 34 working groups
- Well-known: DRMAA, OGSA

Source: ggf.org

GGF: DRMAA

- Batch job submission/control API, capabilities much like Sun Grid Engine
- Specified for C and Java languages
- Implementations: Sun Grid Engine and Project Condor at U. Wisconsin-Madison

Source: DRMAA.org

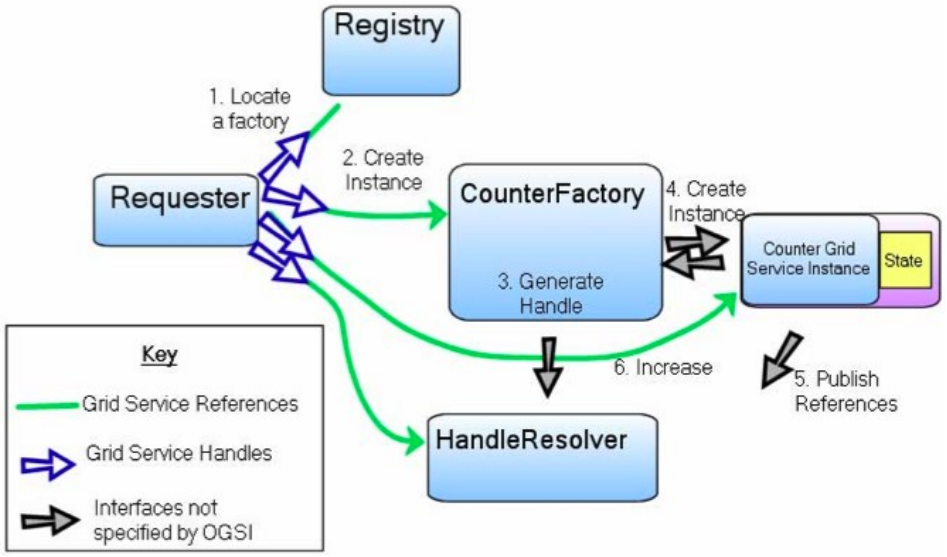
GGF: OGSA

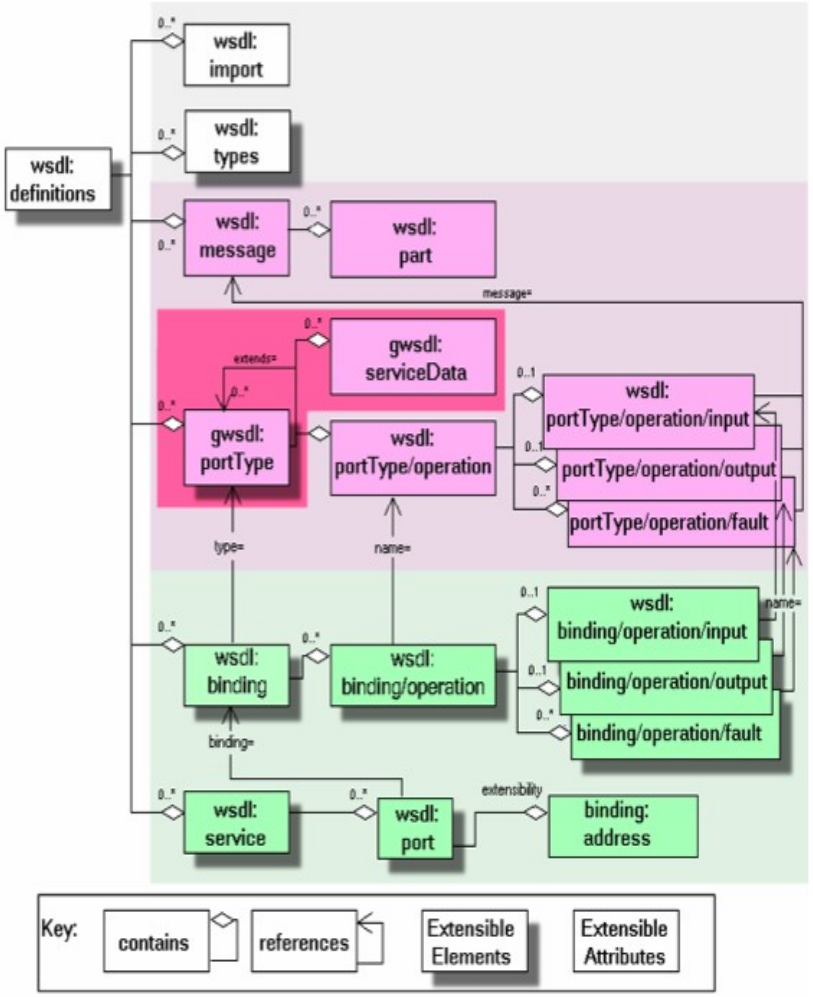
"Open Grid Services Architecture"

- Open, service-oriented grids based on WS-*
- Some use cases: data center management, reactive storm modeling, large-scale archive

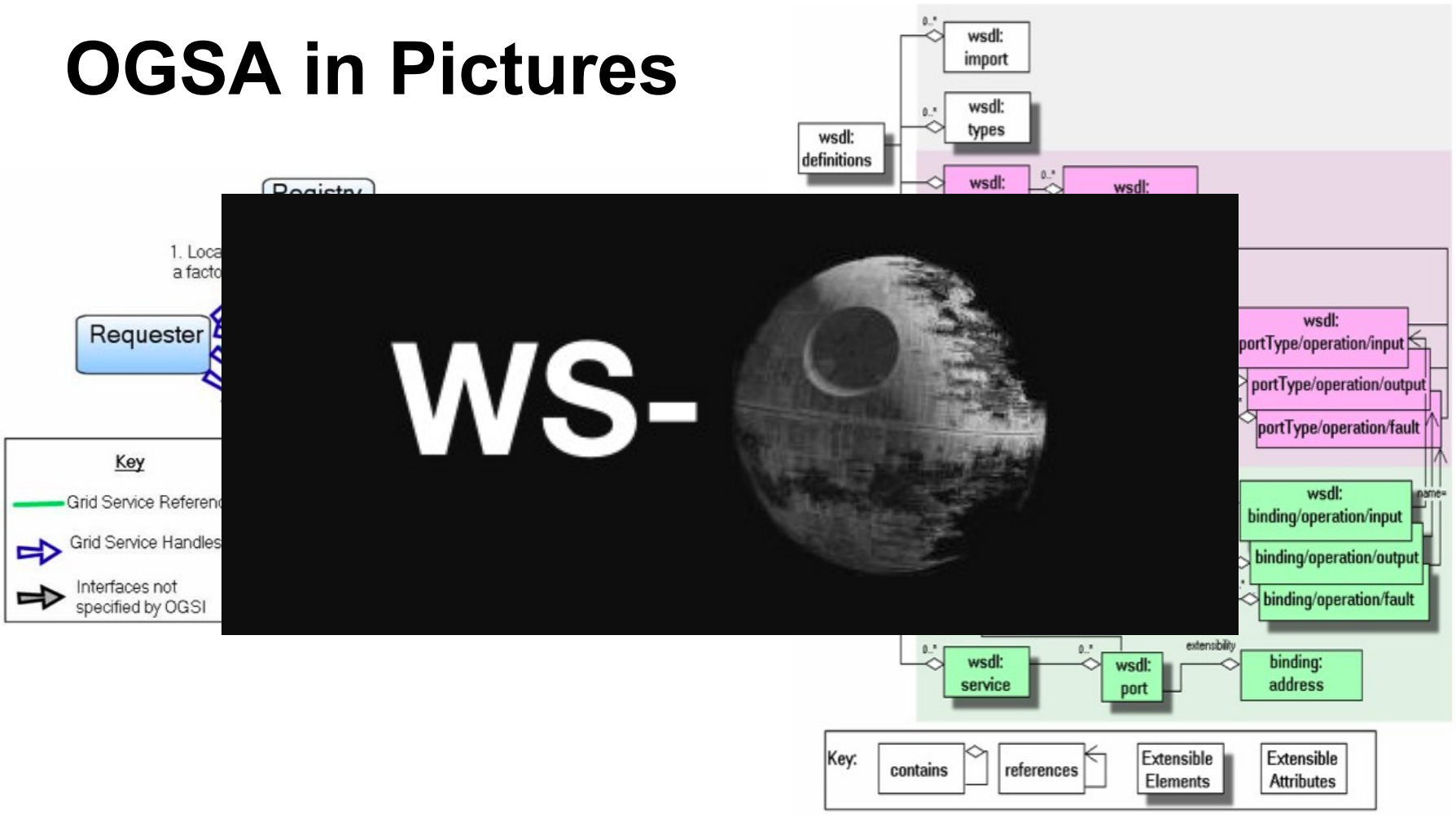
Source (all OGSA): GGF.org

OGSA in Pictures





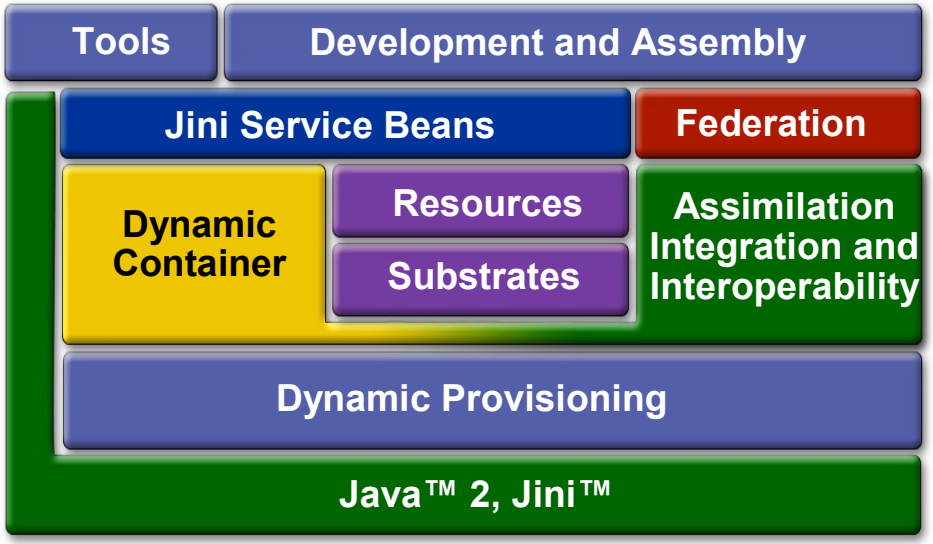
OGSA in Pictures



Source: David Heinemeier Hansson (loudthinking.org/arc/000585.html)

The Jini™ Technology-based Project Rio Framework

- "Project Rio provides a model to dynamically instantiate, monitor and manage service components as described in an architectural meta-model called an OperationalString"
- "Hello World" has one interface, six classes, and 625 lines of Java language code



Source: rio.jini.org

The Gridbus Project

- Led by Rajkumar Buyya of Melbourne U.
- Ambitious framework includes an economic model, SLAs, and a Market Directory
- “Alchemi” .NET-based implementation

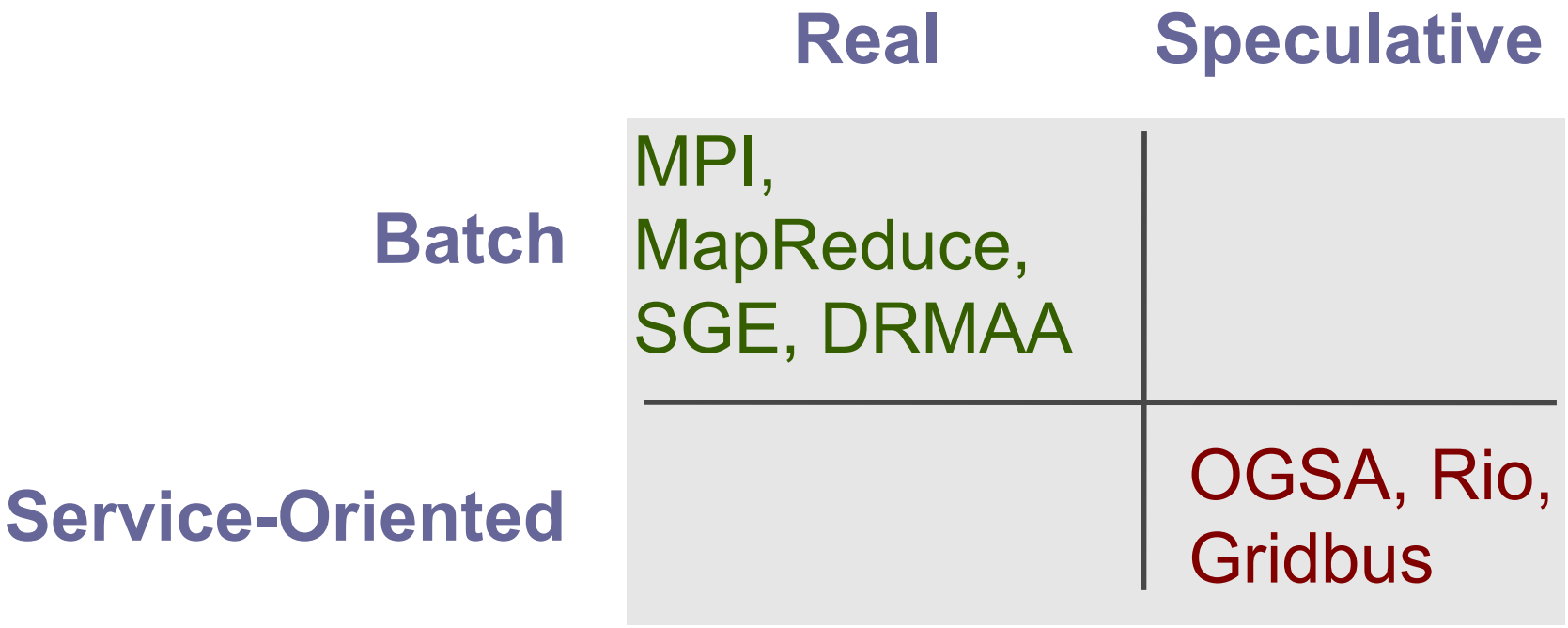
Source: www.gridbus.org

The Grid Landscape

	Real	Speculative
Batch	MPI, MapReduce, SGE, DRMAA	
Service-Oriented	Google, Yahoo, etc.	OGSA, Rio, Gridbus

The Grid Infrastructure Landscape

The Grid Infrastructure Landscape



Agenda

Why Grids? Why Not?

Grid History:
Batch and Service Orientation

Existing Grid Frameworks (How to Say "Hello World" 10^8 Times in Parallel)

Project Sigrid

Project Sigrid's Design Center

AKA the UNIX® + Web world-view

- Tell me how many servers are available
- Find a server and run a program for me on it
- Tell me what's running, allow me to kill it, and notify me when it stops
- Stage modestly sized code and data files
- Assume that my program will listen on a socket and I want to connect to it
- Route input, output, and error streams, assuming they are textual

Launch an Echo Responder

```
Sigrid s = new Sigrid("HelloSigrid");
Task t = new Task(s, "perl-echo", "perl data/echo.pl");
t.addFile("/Users/twbray/dev/pd/tests/demo3.pl",
          "echo.pl", "data");
t.consumeOutput();
t.acceptInput();
t.routeError("10.0.0.33", 9321);
if (s.availableServerCount("perl-echo") > 0) {
  t.launch();
  BufferedReader fromTask = t.getTaskOutputStream();
  PrintStream toTask = t.getTaskInputStream();
  String message = new Double(Math.random()).toString();
  toTask.println(message);
  String response = fromTask.readLine();
}
```

Launch and Find a Task

```
ArrayList<HostPort> monlist = sigrid.nowRunning(MON_TYPE);
if (monlist.size() == 0) {
  sigrid.watchForChanges(this, MON_TYPE);
  Task zepMon = new Task(sigrid, MON_TYPE, MON_CMD);
  synchronized(this) {
    host = zepMon.launch();
    this.wait();
  }
  monlist = sigrid.nowRunning(MON_TYPE);
  if (monlist.size() != 1)
     throw new Exception("Can't start monitor");
}
HostPort hp = monlist.get(0);
host = hp.getHost();
int port = hp.getPort();
// ready to connect now
```

Fill Up a Grid with Java Language Tasks

```
Task scout = new Task(sigrid, scoutType,
                      "java " + scoutArgs);
scout.addFile(jar, myJar(), "java");

int toLaunch = sigrid.availableServerCount(scoutType);
ArrayList<String> runningOn = new ArrayList<String>();
for (int i = 0; i < toLaunch; i++)
  runningOn.add(scout.launch());
```

How Project Sigrid Works (1)

- Add a node to a simple grid by running a “Null task” with the simple grid’s name as argument
- There’s a “Monitor task” always running on one of the nodes in the simple grid
- Each Null discovers a Monitor;
  if it fails, it starts one
- Monitors discover other Monitors, all but one exit

How Project Sigrid Works (2)

- To start, clients have to discover a Monitor
- Clients interact with the Monitor to request task launch/monitor/kill and pipe-fitting
- Nulls actually do the work
- Some special facilities (CLASSPATH setup) for starting Java language tasks

How Project Sigrid Works (3)

- Discovery is done with JXTA™ technology…
- …but all the JXTA technology weirdness is hidden; could possibly be done with zeroconf instead

Project Sigrid Implementation

- 3500 lines of Java language code, 108K jar
- Requires SE 5
- No objects on the wire! Simple text message protocol
- Highly multi-threaded and concurrent
- Tested on weird ad hoc collections of Mac, Solaris™ OS, and Windows boxes; no big grid yet

What Project Sigrid Is for

- I want a real-time blog search engine
- To store the index in memory, I wanted something like “memcached”, an arbitrarily large persistent in-memory HashMap running across as many computers as necessary; this is called Zeppelin
- Project Sigrid is the necessary infrastructure to let a Zeppelin run across a grid-like collection of computers
- Zeppelin runs very fast

Summary

- Grids are useful for many, but not all, problems
- There is lots of batch-oriented grid computing
- There is some batch-oriented grid infrastructure, but not particularly Java technology-friendly
- There is lots of service-oriented grid computing
- There is no service-oriented grid infrastructure
- Project Sigrid is trying to fix that

Q&A

Project Sigrid: The Simplest Possible Grid Computing Platform

Tim Bray (tim.bray@sun.com)

Director of Web Technologies
Sun Microsystems
http://www.tbray.org/ongoing/

TS-3108