Using Condor glide-ins and Parrot to move from Dedicated Resources to the Grid

S. Belforte, M. Norman, S. Sarkar, I. Sfiligoi, D. Thain, F. Wuerthwein

Workshop on Parallel Systems and Algorithms
16 March, 2006, Frankfurt/Main, Germany
Talk Outline

- CDF computing
  - model, resources, requirement
- Embrace Grid computing
  - support existing applications
  - minimize the impact on the end-users
- GlideCAF
  - single- and multi- site flavours
- Software distribution in the Grid environment
  - Parrot over HTTP
- Summary
1. Develop, debug and submit from personal PCs

2. Track your running jobs interactively

3. Satisfied? No need to stay connected anymore

4. Follow your jobs on the web monitor

5. On job completion receive an email

2 central farms in Fermilab
10 distributed farms worldwide

User selects the farm, usually
Fermilab for data analysis
Distributed farms for MC production

CAF
CDF Analysis Farm

Output to any place
CDF Computing – resources and needs

- **Resources:**
  A total of about \(6M\) SPECint2000 with \(\sim50\%\) computing performed outside Fermilab

- **Sustained DAQ rate:** \(60\) MB/s
- **Data used:** \(10-40\) TB/day on-site
- **Job environment**
  - multiuser
  - half the jobs data intensive
  - highly parallelisable

- **Requirement:**
  Current: \(\sim6\) M SPECint2000
  End of 2007: \(\sim15\) M SPECint2000

From 2004 to 2006
- Data logging rate: \(x3\)
- Event rate: \(x4\) due to better compression
CDF Computing - details

Most monitoring via regular web interface
- Both running and completed jobs
- System monitoring, including memory usage and CPU load
- Usage history graph

Supports Interactive monitoring also (dir, ps, tail, top etc.)
- **Schedd** manages user jobs
- **Negotiator** assigns nodes to jobs
- **Starter** manages jobs on the WN
- **Collector** gathers information about other daemons

**Condor System - overview**

- **Schedd**
  - User jobs

- **Negotiator**
  - User priorities

- **Starter**
  - User Job

- **Collector**
  - Gathers information about other daemons

- **UDP Packets**
  - 1
  - 2
  - 3
CDF Computing - must evolve

Expansion of dedicated pools **no more an option**

Turn attention to shared resources

Add dedicated resources to a Grid Pool

- Harvest batch slots using Condor with GSI authentication
  - user jobs pulled onto such slots
- Add a new component to the software framework to manage resources

Additionally, utilize the **idle CPU cycles** at the Grid Site
GlideCAF - overview

Use Condor glide-ins

A simple extension of the CAF (retains all the virtues)

Once a job starts on a WN, it notifies the collector and joins the pool as a new VM

glide-ins are regular, properly configured Condor starter daemons submitted as jobs to the Grid CE

- indistinguishable from a dedicated one to Condor
- will be matched in the same way to a user job with the best priority
- **Negotiator** assigns nodes to jobs
- a secondary schedd manages the glide-ins

Globus assigns jobs to VOs
GlideCAF - deployment

- CNAF-T1, Bologna
  - First prototype: April '05
  - In production: Sept '05
  - ~ 680K jobs processed
  - ~ 200 users
  - Negligible job loss

Use more than your share (~ 400)

Give back when others need it

- San Diego (~300 VMs)
- Fermilab (~600 VMs)
- Lyon-T1 (~100 VMs)
GlideCAF - advantages

- **Late binding of resources**
  - check resource availability in real-time, no need to trust published information
  - only glide-ins wait in – sometimes indefinitely long – queues, but never the user jobs
  - user jobs sent only when resources become available, i.e. after glide-ins started and created VMs

- **Black hole removal**
  - defective nodes kill only glide-ins but no user jobs
  - additional VO specific sanity checks can be performed
  - nodes failing recurrently can be blacklisted
GlideCAF - advantages

- Fine grained policy management with two level negotiation
  - VO level at the Grid site
  - user level in Condor for each VO
- No Grid site specific installation required
  - glide-ins will fetch the binaries required
- Condor only environment
- Only a small incremental change
  - monitoring continues to work natively, only minor modification was needed
Base GlideCAF - limitations

- CDF Software must be available on the WNs through a shared file-system (NFS, AFS)

- Collector and starters need good network
  - UDP based communication will not work over WAN
  - Require bi-directional traffic, will not work over firewalls

- A GlideCAF must be installed at each Grid site
  - reasonable for Large, CDF Friendly sites
    - CNAF, San Diego, Lyon
  - unmanageable if accessing scores of small sites

→ GlideCAFs must be extended
GlideCAF + GCB

GCB is a proxy server, used by the starters to route all the communications

- a starter establishes a persistent TCP connection with the GCB server
- communicates the GCB node and the port obtained to the other interested parties
- from that point on, any Condor daemon can connect via TCP to the specified GCB, that will route the traffic to and from the correct open TCP connection
GCB is Flexible

- A **GCB** server can be set up anywhere in the world, as long as it accepts traffic.
- Several servers can be installed on different machines for scalability.
- **GCB** supports *Strong authentication*.
- For sites which allow neither in-bound nor out-bound traffic, a **GCB** server can be installed on the site border.
- When outgoing connections are available, **GCB** can be used just for notification, and the starters will directly connect to the other daemons using **TCP**, for efficiency.
OSG-CAF – bird's eye view

A single CDF portal for OSG
GlideCAF + GCB + SQUID = OSG-CAF

OSG-CAF is in advanced Beta test
Grid Environment - software distribution

- In the Grid environment, how to access the software distribution?
  - installing a copy at each and every site is unmanageable

- Need a central repository with seamless access to the software distribution(s)
  - logically like NFS, but over WAN
  - AFS could be ideal, but it's a site choice

- Enter Parrot over HTTP
  - access remote data sources as if they are local
  - user level application, no system installation required at Grid sites
Parrot

Ordinary Program

User Defined Namespace
/home/cdfsoft = /httpfs/cdfsoft.fnal.gov/base
/tmp = DENY

Intercepts I/O calls
Name Resolution and Security Policies

The Parrot Virtual File System
RFIO NeST Chirp UNIX FTP HTTP

(PTrace trap)

(POSIX Interface)

Talks several protocols

Partial File I/O (open, close, read, write, lseek)
Whole File I/O (get/put)

Integration with Castor
Allocation and Mgmt
Full UNIX Semantics
Traditional I/O Services

RFIO Server
NeST Server
Chirp Server
Local Disk
FTP Server

HTTP Proxy
HTTP Server

CDF Software

Wide-Area HTTP

Parrot over HTTP-FS

User level application

Integration with Castor
Traditional I/O Services

Local Cache

User level application
Parrot and CDF

- **Grid site**
  - Starter
  - Parrot binary
  - Startup script
    - Config script
      - CDFSOFT script
      - Analysis bin
      - Final cleanup
  - Grid node
    - Local Disk
    - SQUID Cache
    - Central Repository
- **Submitter**
  - Condor
    - Schedd
- **CDF SOFT**
  - HTTPd
  - **Central Repository**
Parrot - performance

Local I/O with Parrot binaries

<table>
<thead>
<tr>
<th>Throughput</th>
<th>Traps/s</th>
<th>Unmodified</th>
<th>w/Parrot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MB/s</td>
<td>15k</td>
<td>199s</td>
<td>199s</td>
</tr>
<tr>
<td>8 MB/s</td>
<td>120k</td>
<td>199s</td>
<td>199s</td>
</tr>
<tr>
<td>20 MB/s</td>
<td>300k</td>
<td>201s</td>
<td>205s</td>
</tr>
<tr>
<td>40 MB/s</td>
<td>600k</td>
<td>204s</td>
<td>215s</td>
</tr>
</tbody>
</table>

- no performance penalty upto 8 MB/s
  - typical for CDF application needs

- at 40 MB/s gets hit by 5%
**Parrot - performance**

<table>
<thead>
<tr>
<th>CDF Application</th>
<th>Unmodified</th>
<th>w/Parrot</th>
<th>w/HTTPFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real simulation</td>
<td>11.9h</td>
<td>12.1h</td>
<td>12.5h</td>
</tr>
<tr>
<td>Toy Monte Carlo</td>
<td>16.6m</td>
<td>16.8m</td>
<td>17.3m</td>
</tr>
</tbody>
</table>

- about 5% slowdown for realistic jobs using remote FS in PATH
- achieved with SQUID cache
- since Parrot over HTTP will harness more CPUs, overall through-put will certainly increase

Parrot in CDF is in Beta test
Summary

• CDF is able to use Grid resources
  – a number of base GlideCAF\textregistered\textsubscript{s} in production

• Software infrastructure preserved
  – transparent to end-users; monitoring continues to work in a standard manner

• No new middle-ware introduced
  – just using Condor differently
  – ensured a rapid turn-around

• Future direction
  – GlideCAF\textregistered\textsubscript{s} extensions + Parrot over HTTP for software distribution
Backup Slides
Interactive Monitoring

Only outgoing connections from desktop

Command Line
- list of jobs
- process list of a section
- listing the working directory
- tail
- debugging a process
Condor System - Condor-G

Monitoring would need re-implementation

- Globus assigns nodes to jobs
- All control at the Grid site
GlideCAF Details - glide-ins

GSI authentication all along
The Web Server distributes glide-in binaries and site specific configuration to WNs

A single GSI CDF service proxy used for all the glide-ins
GlideCAF issues

- A single Grid certificate used for all the glide-ins
  - The site gatekeeper finds only one special user for that VO
    - no knowledge about the real users
- Looking at future evolutions of glExec for a solution
  - gets the GSI proxy and authenticates the real user
  - possibly changes the UID
- Real users authorized prior to job execution
- Sites can blacklist certain users
Parrot Performance Benchmark

System Call Latency

- getpid
- stat
- open-close
- read 1B
- read 8KB

Time (us)