Managing jobs with an interpreted language for dynamic adaptation

{anolan, noemi}@inf.puc-rio,
schulze@lncc.br

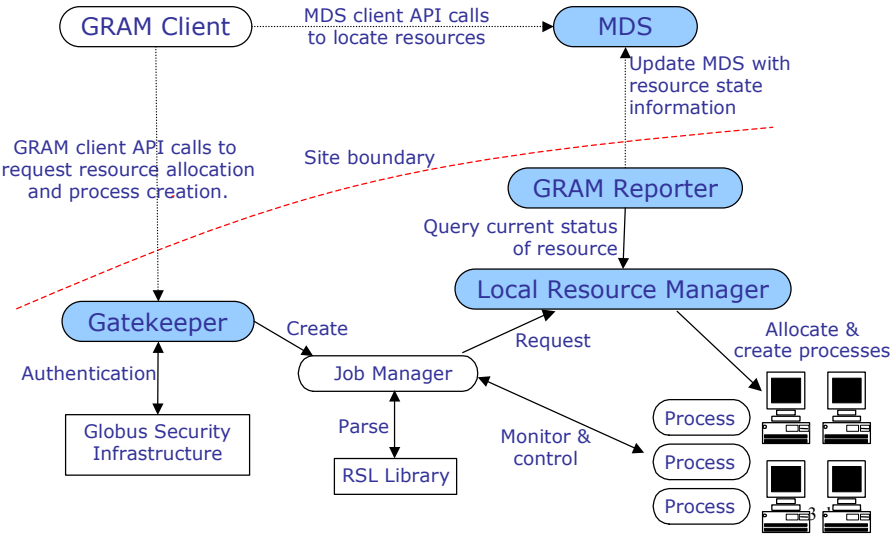
Laboratório Nacional de Computação Científica

Motivation

- To develop a light engine for the allocation of resources for computational jobs submitted through the Grid.
  - Clusters with virtual addressing

Laboratório Nacional de Computação Científica

Globus GRAM Components



Local Resource Manager

- LSF (Load Sharing Facility)
- SGE (Sun Grid Engine)
- OpenPBS (Open Portable Batch System)
- Condor

Common features

- Checkpointing and job migration
- Dynamic Load Balancing
- Authentication and authorization
- Daemon fault recovery
- Staging

Grid characteristics

- Administrative boundaries
- High variability of resource utilization
- User Interaction
  - Long life applications can profit from user actions along the execution
- High failure probability

Dual-language approach

Compiled Languages

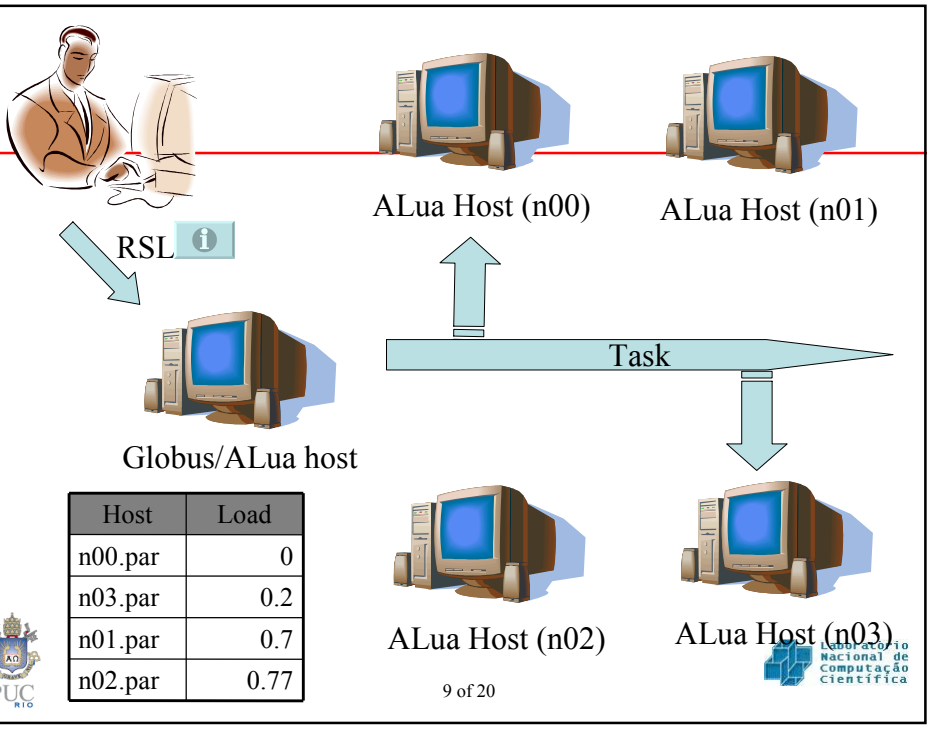
- High performance computing
- Security

Interpreted Languages

- Portability
- Flexibility
- Interactivity: functions can be redefined or added dynamically

ALua

- Event based communication mechanism for parallel and distributed programming in Lua.
- ALua applications are composed by various processes executing in different machines that communicate through messages.
- The messages are code chunks that are executed by the receiver.
- Code may contain parts in C.
- Concurrence
  - Single thread. Messages are sequentially executed by the event loop.

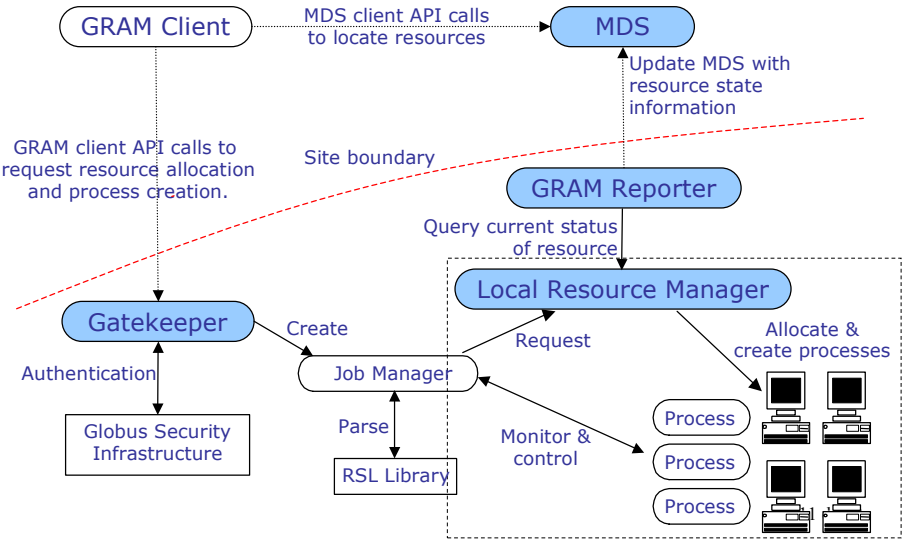


Host	Load
n00.par	0
n03.par	0.2
n01.par	0.7
n02.par	0.77

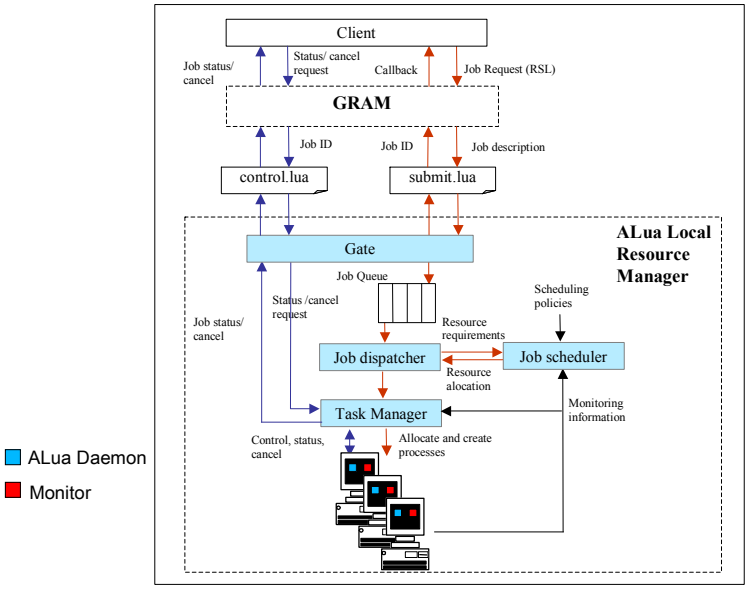
Grid submission

- globusrun -r server.par.inf.puc-rio.br:/jobmanager-alua '&(executable=ls)(count=2)'

Globus GRAM Components



Local Resource Manager



Notification engine (Lua Monitor)

- Extensible monitoring system written in Lua
- Based on
  - Properties: parameters to be observed
  - Aspects: allow programmer to observe specific behaviour of the property
  - Observers: specify the properties/aspects of interest and a callback for receiving notifications when an event occurs

Lua Monitor

- Example:

```
LuaMonitor:attachEventObserver({
notifyEvent=function(self, event)
    alua.send(taskManager,[[alarm()]])},
"CPUIncrease","$CPU:Increasing")
```

Grid submission

- globusrun -r server.par:/jobmanager-alua '&(executable=ver00) (count=1) (jobType="mpi") (arguments="11 1 0 0 0 285615")'
  globus_gram_client_callback_allow successful
  GRAM Job submission successful
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_ACTIVE
  GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE

Performance results*

Instance	Remote execution	ALua	ALua with monitors
nl14	295.34	295.56	295.72
nl16	2291.54	2293.668	2295.352

* Experiments performed on a set of Intel Pentium II CPU with 398 MHz and 280 of RAM

Console

```
print([[active_jobs()]])
Answer: 1

define([[manager.active_jobs =
  manager.pending_jobs]]
print([[manager.active_jobs()]])
Answer: 0
```

Conclusions

- Light and effective engine for job submitting
- Monitoring and interactivity
- A lot of work to do!

Future work

- Resolve security problems
- Tools to allow for interaction (console)
- Allow processes in different sites to communicate
- Meta-scheduling
- Performance tests with real applications

More information

- Portal Giga
  http://giga01.lncc.br:8080
- http://alua.inf.puc-rio.br

The End