Multi-Site Jobs Management System (MJMS): A tool to manage multi-site MPI applications execution in Grid environment

Jaime Frey[1], Francesco Gregoretti[2], Giuliano Laccetti[3], Almerico Murli[3], Gennaro Oliva[2]

Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System
- MJMS Architecture
  - MJMS Server, Coordinator, GangMatchMaker, Advertiser
- Multi-Site Job Life Cycle
- Preliminary experience
- Future Works

Outline

- Multi-site Parallel Application

Multi-Site Parallel Application

- Multiple distributed processes running on one or more computing resources in a Grid environment:
  - Heterogeneous
  - High-Performance
  - Geographically distributed

MPI on the Grid

- The Message Passing Interface standard is:
  - easy to understand and use
  - architecture-independent
  - portable
  - widely-used
- MPI eases Grid applications development to programmers with parallel computing skills

Grid-enabled MPI implementations

- MagPIe
- MPICH-G2
- MPI Connect
- MetaMPICH
- Stampi
- PACX-MPI

Grid-enabled MPI implementations

- MagPIe
- MPICH-G2
- MPI Connect
- MetaMPICH
- Stampi
- PACX-MPI

- group multiple Grid resources potentially heterogeneous for the execution of MPI programs
- use vendor-supplied MPI libraries over high-performance networks for intra-machine messaging

Missing Application Management System

- Advanced libraries support coallocation an synchronization
- No one provides advanced execution management tools
- Users must explicitly:
  - specify resources to be used for the execution
  - directly handle possible failures

Outline

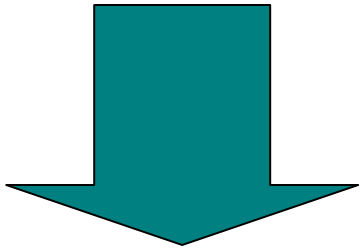
- Multi-site Parallel Application
- Grid Application Management System

Grid Job Management Systems

- The adoption of job management systems (Condor-G, Datagrid WMS) facilitates the use of the Grid for sequential and parallel applications
- These systems can transparently handle on the users behalf:
  - resource selection
  - resource allocation
  - file transfers
  - application monitoring
  - failure events

Grid Job Management Systems

- Existing management systems don't handle requests for multiple resources within a single job



- They don't allow the execution of multi-site parallel applications

Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System

MPI Job Management System

- MJMS manages the execution of multi-site parallel MPI applications in a Grid environment
- MJMS
  - interacts with the Condor-G daemons for job execution management
  - uses the services provided by DUROC to handle job synchronization across sites

Condor-G features

- Interacts with the GRAM service for job submission and monitoring
- Provides a high-level language called ClassAds to describe job requirements and preferences and Grid resources characteristics
- Has some fault-tolerance features
- Can manage distributed I/O operations
- Provides a robust mechanism called Matchmaking to match a job request with a computing resource

Condor-G extensions

- Condor-G system has been extended in order to support multi-site parallel applications
  - Support for MPICH-G2 grid enabled MPI implementation
  - Substitute Matchmaking mechanism is limited to match each single job request with a single resource
  - the condor_collector daemon doesn't interact with the Globus Information System

MPI Job Management System

- MJMS allows the effective, reliable and secure execution of multi-site parallel applications in the Grid environment
  - Based on Condor-G
  - Uses Condor submit description file syntax
  - …
  - Selects available computing resources according to application requirements
  - Interacts with the local management systems through Condor-G

Users must

- User splits her application into subjobs
- Each subjob is assigned to a Grid resource
- User can specify requirements and preferences for each subjob
- Requirements and preferences reflect needs in terms of computational and communication costs
- Subjobs can have …interdependecies…

MJMS features

- MJMS assigns each subjob to a single Grid resource
- MJMS locates a matching pool of available resources according to the application requirements and preferences and subjobs interdependencies
- Resources are picked in the Globus Information System

(subjob1)
universe = globus
executable = bcg_dist
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
requirements = (OpSys == ‘‘LINUX’’ && Arch == ‘‘i686’’) && (ClusterNodeCount >= 17)
environment = P4_SETS_ALL_ENVVARS=1; MGF_PRIVATE_SLAN=1;
rank = 10000/ClusterNetLatency + 10*ClusterCPUsSpecFloat
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
globusrsl = (jobtype=mpi) (count=17) (label=subjob 0)
LD_LIBRARY_PATH=/opt/globus-2.4.3/lib/
when_to_transfer_output = ON_EXIT
should_transfer_files = YES
output = outfile.$(Cluster)
error = errfile.$(Cluster)
log = logfile.$(Cluster)
machine_count = 17
queue

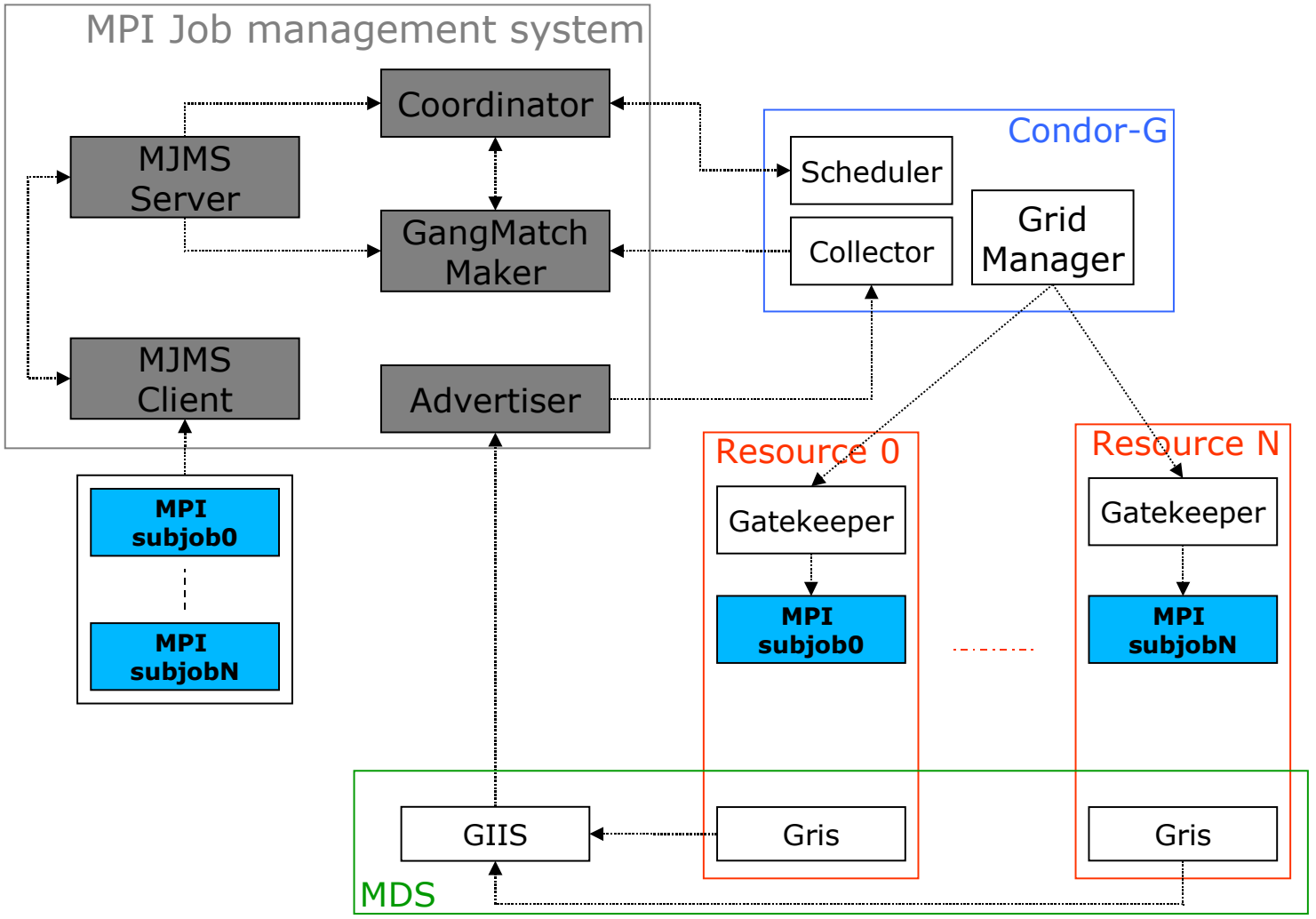
(subjob2)
universe = globus
executable = bcg_dist
machine_count = 4
log = logfile.$(Cluster)
error = errfile.$(Cluster)
output = outfile.$(Cluster)
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
LD_LIBRARY_PATH=/opt/globus-2.4.3/lib/
globusrsl = (jobtype=mpi) (count=4) (label=subjob 17)
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
environment = P4_SETS_ALL_ENVVARS=1; MGF_PRIVATE_SLAN=1;
requirements = (OpSys == ‘‘LINUX’’ && Arch == ‘‘i686’’) && (ClusterNodeCount >= 4)
queue

(subjob3)
universe = globus
machine_count = 14
executable = bcg_dist
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
requirements = (OpSys == ‘‘LINUX’’ && Arch == ‘‘i686’’) && ClusterNodeCount >= 14)
environment = P4_SETS_ALL_ENVVARS=1; MGF_PRIVATE_SLAN=1;
globusrsl = (jobtype=mpi) (count=14) (label=subjob 21)
LD_LIBRARY_PATH=/opt/globus-2.4.3/lib/
rank = 10*ClusterCPUsSpecFloat
+Subnet1 = subjob1.Subnet
output = outfile.$(Cluster)
error = errfile.$(Cluster)
log = logfile.$(Cluster)
queue

Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System
- MJMS Architecture
  - MJMS Server, Coordinator, GangMatchMaker, Advertiser

MJMS Architecture



MJMS Server

- Accepts submission and cancellation requests and queries about job status
- Executes and coordinates the other components
- User communicates with the server by using the client commands:
  - mjms_submit
  - mjms_status
  - mjms_cancel

Coordinator

- Interacts with the *condor_scheduler* for job scheduling
- Uses the DUROC services for subjobs synchronization
- Queries the GangMatchMaker for a pool of consistent resources

GangMatchMaker

- The GangMatchMaker uses the Gangmatching model to locate a pool of resources for multi-site job execution
- The GangMatchMaker is able to match the subjobs execution requests and the available resources by simultaneously taking into account:
  - Subjobs requirements
  - Subjobs preferences
  - Subjob interdependencies

Advertiser

- The Advertiser periodically sends ClassAds representing available resources status and characteristics to the *condor_collector*
- The Advertiser gathers information by querying a set of Globus MDS servers, extracting data relevant to MJMS

Advertiser

- Parallel and multi-site parallel application generally have different requirements from those of sequential applications
  - high-performance network characteristics
  - resource neighborhood
- The Globus Information System have been extended with a custom schema and a corresponding Information Provider in order to publish resources network characteristics

Advertiser

- Information are extracted by the Advertiser and can be used by the GangMatchMaker when performing matches
- User is therefore able to specify conditional expressions with respect to the network performances characteristics

```
system-network-vendor: Quadrics
cluster-network-model: Elan
cluster-network-version: 4
cluster-network-mpi-0-latency: 2.63
cluster-network-mpi-1024-latency: 6.32
cluster-network-mpi-32768-latency: 44.40
cluster-network-mpi-64-bandwidth: 25.64
cluster-network-mpi-1024-bandwidth: 177.95
cluster-network-mpi-32768-bandwidth: 718.91
```

Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System
- MJMS Architecture
  - MJMS Server, Coordinator, GangMatchMaker, Advertiser
- Multi-Site Job Life Cycle

Job Lifecycle (1/4)

- User submits her job to the MJMS Server
- Job Status becomes ACCEPTED
- The server submits execution request to the Coordinator
- Coordinator queues subjobs in the *condor_scheduler* in hold state
- Coordinator queries the GangMatchMaker for a pool of resources

Job Lifecycle (2/4)

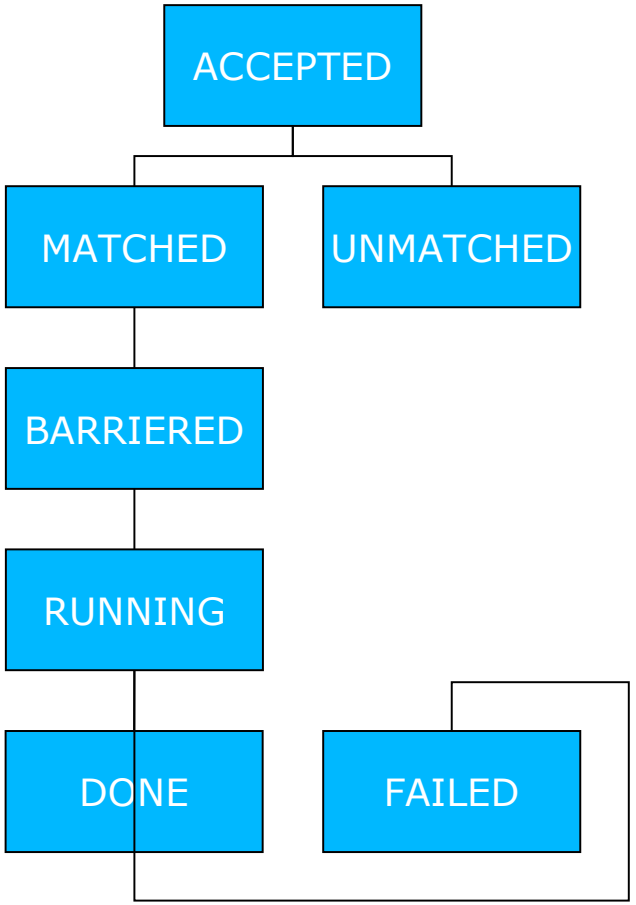
- The GandMatchMaker queries the *condor_collector* for available resources
- If GMM finds resources that fits job needs, it sends their **globus contact strings** to the Coordinator and the JOB status becomes MATCHED
- If such resources are not available the JOB status becomes UNMATCHED

Job Lifecycle (3/4)

- Coordinator modify the target of the subjbos in the Condor queue and release them
- From this point Condor-G takes care of job execution
  - Contacts the Gram Manager
  - Schedule subjobs execution on the local management system
- Once a subjob have been placed in execution all its processes stops at the DUROC barrier

Job Lifecycle (4/4)

- Once all the subjobs arrive to the DUROC barrier computation starts
- JOB state becomes RUNNING



Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System
- MJMS Architecture
  - MJMS Server, Coordinator, GangMatchMaker, Advertiser
- Multi-Site Job Life Cycle
- Preliminary experience

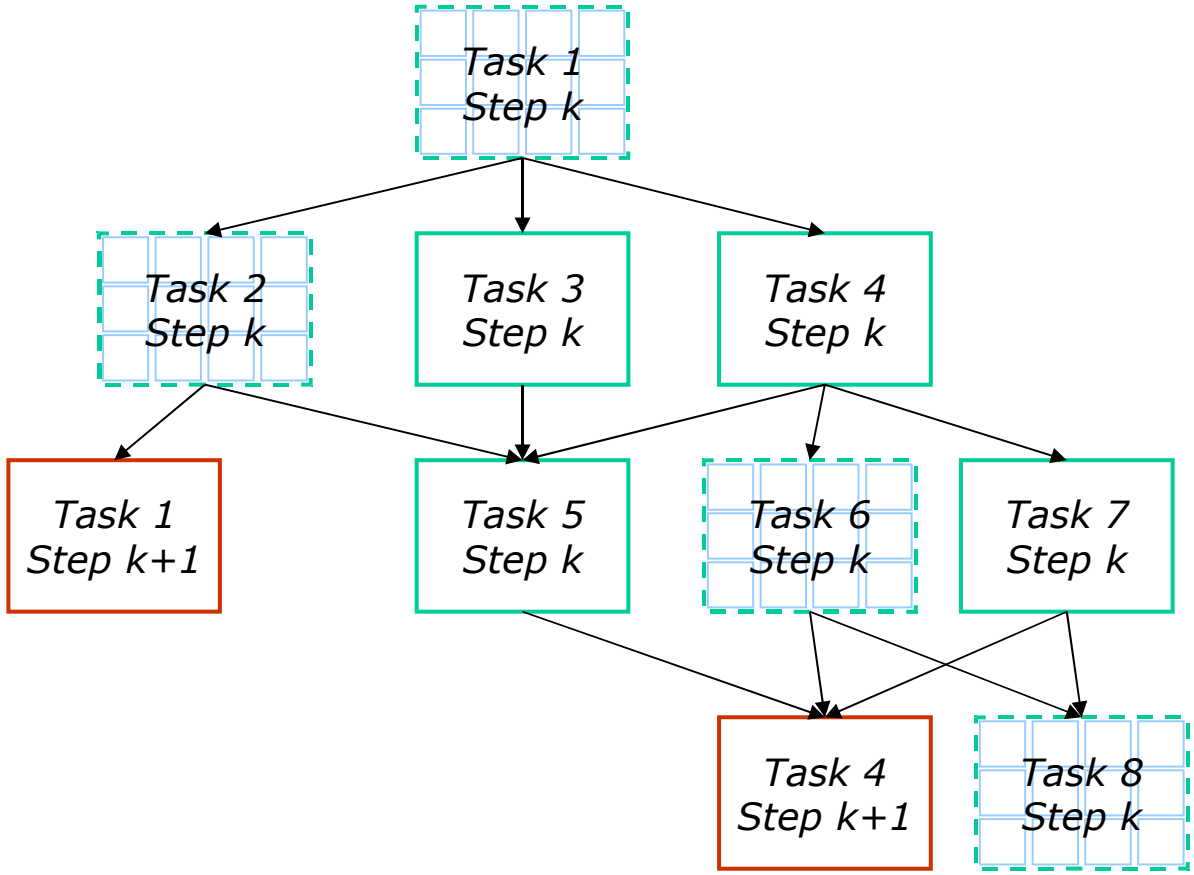
Preliminary experience

- MJMS was used to run an iterative solver of sparse linear systems of equations with multiple right-hand sides in the Grid environment.
- In this context the Block version of the Conjugate Gradient algorithm (BCG) becomes attractive: it reduces the number of synchronization points and increases the size of messages improving latency tolerance

BCG Algorithm

- The BCG algorithm consists of eight concurrent tasks with different computational complexity
- To correctly balance the computational load we introduced a two-level parallelism schema in its implementation

BCG Dependencies DAG



BCG Algorithm

- First level reflects task decomposition
- Second level has been introduced within the most computationally intensive tasks
- The resulting multi-site job consists of different subjobs each grouping one or more tasks
- Each subjob must be assigned to a computing resource

BCG Algorithm

- BCG implementation uses a grid-enabled MPI library based on MPICH-G2 called MGF
- MGF allows transparent and efficient usage of Grid resources including clusters with private networks
- MJMS assigns the subjobs to the available parallel computing resources according to their computational costs and inter-task communication costs
- MJMS let us to perform a better resource workload balancing that improved the overall speedup performance.

BCG subjobs Decomposition

(subjob1) → task 1, 2

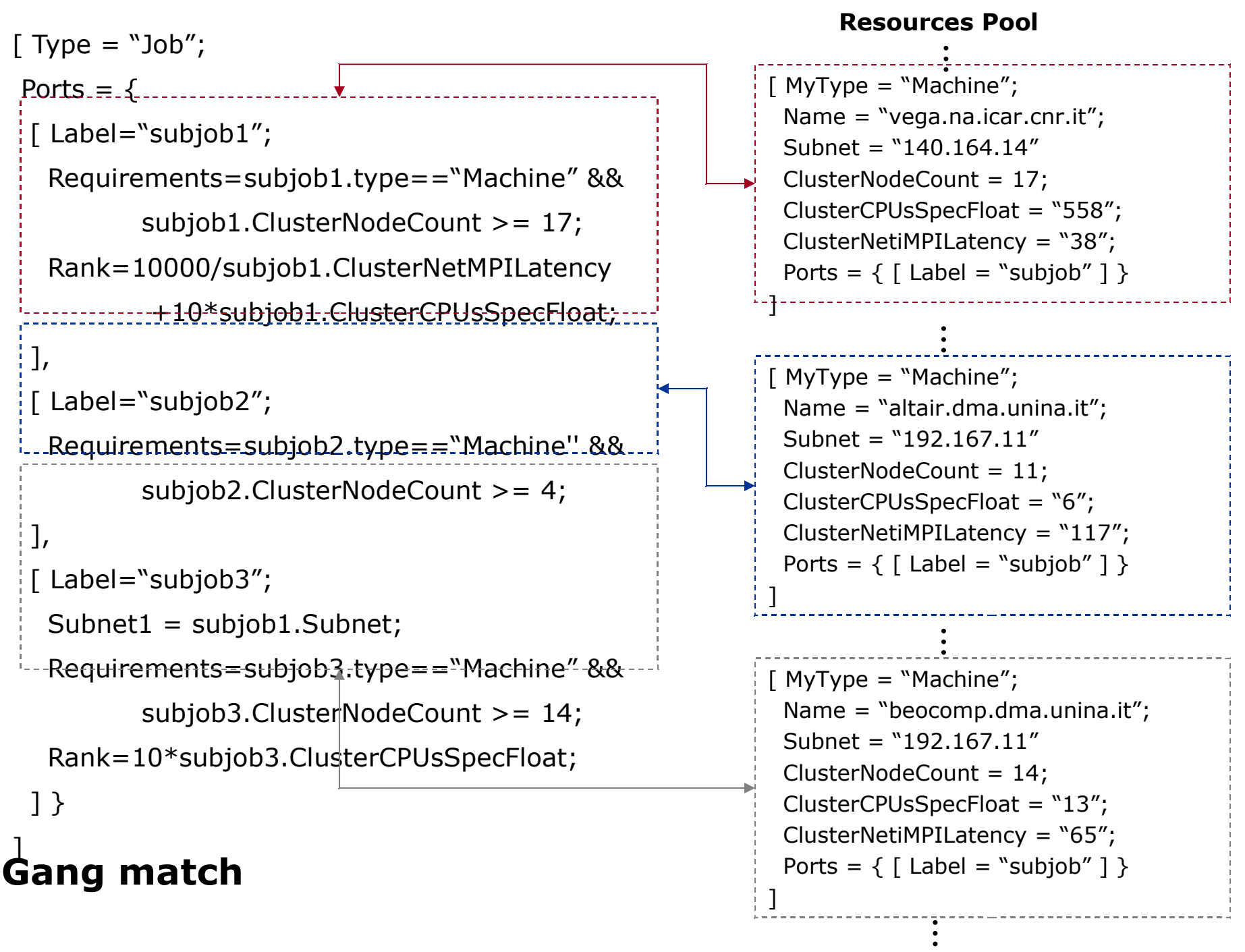
```
universe = globus
executable = bcg_dist
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
machine_count = 17
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
output = outfile.$(Cluster)
globusrsl = (jobtype=mpi) (count=17) (label=subjob 0)
requirements = (OpSys == ``LINUX'' && Arch == ``i686'') && (ClusterNodeCount >= 17)
rank = 10000/ClusterNetLatency + 10*ClusterCPUsSpecFloat
```

(subjob2) → task 3, 4, 5

```
universe = globus
executable = bcg_dist
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
machine_count = 4
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
output = outfile.$(Cluster)
globusrsl = (jobtype=mpi) (count=4) (label=subjob 17)
requirements = (OpSys == ``LINUX'' && Arch == ``i686'') && (ClusterNodeCount >= 4)
```

(subjob3) → task 6, 7, 8

```
universe = globus
executable = bcg_dist
arguments = s3rmt3m3.mtx 3 bs3rmt3m3.mtx 7 15 16 17 18 24 25 31
machine_count = 14
transfer_input_files = s3rmt3m3.mtx,s3rmt3m3_rhs.mtx
output = outfile.$(Cluster)
globusrsl = (jobtype=mpi) (count=14) (label=subjob 21)
requirements = (OpSys == ``LINUX'' && Arch == ``i686'') && (ClusterNodeCount >= 14)
rank = 10*ClusterCPUsSpecFloat
```


[Type = "Job";
Ports = {
[Label="subjob1";
Requirements=subjob1.type=="Machine" &&
subjob1.ClusterNodeCount >= 17;
Rank=10000/subjob1.ClusterNetMPILatency
+10*subjob1.ClusterCPUsSpecFloat;
],
[Label="subjob2";
Requirements=subjob2.type=="Machine" &&
subjob2.ClusterNodeCount >= 4;
],
[Label="subjob3";
Subnet1 = subjob1.Subnet;
Requirements=subjob3.type=="Machine" &&
subjob3.ClusterNodeCount >= 14;
Rank=10*subjob3.ClusterCPUsSpecFloat;
] }
]
Gang match
Resources Pool
[MyType = "Machine";
Name = "vega.na.icar.cnr.it";
Subnet = "140.164.14"
ClusterNodeCount = 17;
ClusterCPUsSpecFloat = "558";
ClusterNetiMPILatency = "38";
Ports = { [Label = "subjob"] }
]
[MyType = "Machine";
Name = "altair.dma.unina.it";
Subnet = "192.167.11"
ClusterNodeCount = 11;
ClusterCPUsSpecFloat = "6";
ClusterNetiMPILatency = "117";
Ports = { [Label = "subjob"] }
]
[MyType = "Machine";
Name = "beocomp.dma.unina.it";
Subnet = "192.167.11"
ClusterNodeCount = 14;
ClusterCPUsSpecFloat = "13";
ClusterNetiMPILatency = "65";
Ports = { [Label = "subjob"] }
]

Outline

- Multi-site Parallel Application
- Grid Application Management System
- MJMS System
- MJMS Architecture
  - MJMS Server, Coordinator, GangMatchMaker, Advertiser
- Multi-Site Job Life Cycle
- Preliminary experience
- Future Works

Future: Multi-institution Multi-user Environment

- MJMS was designed on top of the Condor-G System a personal desktop agent but can be used in multi-institution multi-user environment
- The DataGrid WorkLoad Management System based on the Condor-G system, is used in the EGEE/LCG production Grid by thousands of users and hundreds of institutions
- MJMS can be used in a multi-user and multi-institutional environment by manging resource selection on the basis of Virtual Organization (VO) membership with a system to verify user VO association like VOMS

Future: Local batch management interaction

- Another issue for the use of MJMS in a production Grid is the interaction with the local batch management systems
- The GangMatchMaker should take care of the batch systems queues when matching subjobs with resources
- Queues characteristics and status can be published by the Globus Information System and considered when locating a computing resource for job execution
- Resource selection becomes hard when there are not enough free resources available and some subjobs need to be queued on the local batch systems before running

Future: MPI implementation and Grid Techonologies

- MJMS implementation has been based on MPICH-G2 and pre-WS GRAM
- MJMS design is independent of the MPI implementation and Grid technology
- MJMS implementation can be extended to support other grid-enabled MPI libraries (e.g. PACX-MPI) and Grid technologies supported by Condor-G:
  - NorduGrid
  - Unicore
  - WS GRAM
  - Remote Condor pools

Future:Aggregate Matching

- Job descriptions have to specify the number of subjobs and the number of CPUs in each subjob.
- GangMatchMaker should allow an aggregate matching policy to increase scheduling flexibility
- A job asks for a total number CPUs, resources advertise some number of currently available CPUs, and the GangMatchMaker adds resources to the gang until the number of CPUs requested is reached