MPI Working Group update

D. H. van Dok

EGEE-III INFSO-RI-222667 SA3 All Hands Meeting, Nicosia, 7–8 May 2009

1 Introduction

Many (scientific) problems lend themselves to a parallel computing, where the problem is divided into many sub-problems, each of which can be solved on a single computer in a reasonable time frame.

Berkeley's 'seven dwarfs'
Seven classes of parallel computing methods.

1. dense linear algebra
2. sparse linear algebra
3. spectral methods
4. n-body methods
5. structured grids
6. unstructured grids
7. monte carlo

According to the Berkeley classification [1] these problems can be classified according to the kind of computations that are involved, and hence the kind and intensity of communication required between the computers. This ranges from very communication-intensive to almost no communication, which is called embarrassingly parallel.

The kind of computational resource used to address such scientific problems is usually chosen to make the most efficient use of it. To oversimplify it a little, the communication-intense problems are best run on a supercomputer with fast, low-latency networks between the nodes, and embarrassingly parallel jobs are run on cheaper hardware, such as generic clusters, desktop clusters, home computer clusters or grids.

Grids, as the traditional view holds, is mainly for running lots of independent jobs.

But why? There are science applications that would use the grid for parallel programming, and there are resource providers that have specialized hardware for the job. And even on more generic cluster systems, parallel programming systems such as MPI or OpenMP could be used that would benefit some categories of applications.

2 Problem identification

What seems to be the problem

1. Not enough resources available for parallel computing
2. not enough users run their parallel problems on the grid

The general impression is that (1) there are not enough resources available to offer parallel computing (MPI) and (2) that not enough users who could potentially make use of such resources actually run their problems in parallel on the grid, for whatever reason.

The question is whether this impression is correct. There is at least some evidence for the first case: we can simply enumerate the grid resources by exploring the information system, and find that only a fraction of clusters advertise the availability of MPICH, OpenMPI or suchlike.

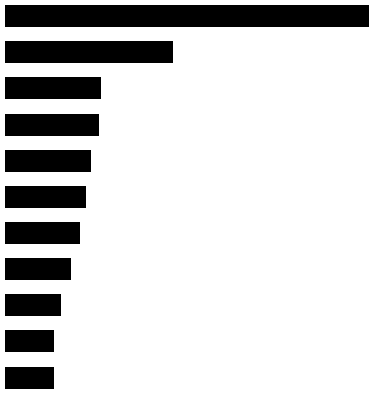
findings from the information system

About **126** out of **484** listed SubClusters list anything like 'mpi' in the `GlueHostApplicationSoftwa` or roughly **25%.**

The most frequently reported tags:

MPICH	171
MPI-START	79
MPICH-1.2.7p1	45
OPENMPI	44
MPICH-1.2.7	40
MPI_SHARED_HOME	38
MPICH2	35
MPI_NO_SHARED_HOME	31
MPICH2-1.0.4	26
VO-balticgrid-E-PARALLEL-MPICH2	23
OPENMPI-1.1	23

(65 more)



- SAM tests were discontinued due to too many failures
- supporting software is out of date, not maintained?

Another giveaway is on the operational level; there used to be a SAM test for MPI sites, but these were discontinued because too many sites simply failed the basic tests. The instructions for site administrators to set up MPI are not really hard, but neither are they straightforward and the state of the support for helper software is unclear.

User Survey

Questions to the users:

- Can you give an estimate how intense the processing is, in terms of total CPU hours and the period over which these were used?
- Where have you found resources to run your programs in the past?
- Please estimate the size of previously used resource provider (total CPUs)
- Are you using the Grid to do multi-core computations, using MPI, OpenMP or similar techniques?
- Did you ask the Grid sites for support?
- Did you seek help in the EGEE community?
- What were the answers you got?
- Did you have to convince the system operators to install MPI?
- Was it easy to find documentation for running in parallel on the Grid?
- How difficult would you rate doing parallel computations on the Grid?

To help bring some clarity on the user's situation, the MPI working group has held a survey among potential users, asking whether they had considered using the grid, had in fact tried and had in fact failed, and why. The findings are currently being collected but a preliminary impression is that users do encounter difficulties, but there is not a single common fix that will make it work for everybody.

3 Many small solutions

The approach to ameliorate the situation is therefore manifold, and the recommendations of the MPI WG reflect this.

In the following, the full problem area is explored in the 'natural' order as a user would encounter it. In the end, the preliminary recommendations of the working group are summarized.

A virtual tour of a parallel job

A user would typically visit the following waypoints:

- discovery of resources
- matchmaking
- (porting the application)
- passing job requirements
- scheduling
- initializing the system

- running and collecting output

Submitting parallel jobs begins with a search for available resources. The ultimate source of information would be the all-encompassing information system (IS), although realistically many users would, through their virtual organization membership, only have a moderate set of resources to choose from and know in advance which clusters fit the bill.

3.1 publishing the right tags

In any case, relying on the IS has its issues. For starters, there seems to be little consensus about what exactly to publish.

Publishing tags in the IS

The new recommendation tentatively states that

```
GlueHostApplicationSoftwareRunTimeEnvironment: Parallel
GlueHostApplicationSoftwareRunTimeEnvironment: ⟨flavour⟩-⟨version⟩
GlueHostApplicationSoftwareRunTimeEnvironment: ⟨flavour⟩
```

should be used. Perhaps we should establish the canonical names for each possible flavour as well, to prevent varieties like OpenMPI, OPENMPI and OPEN-MPI from cropping up.

The matchmaking process will match the job requirements to those resources that have the right tag; unfortunately you can only specify an exact version, so matchmaking a la 'greater-than-or-equal' is not possible.

3.2 scheduling to match desired granularity

The classical "Jobtype=MPICH" should no longer be necessary, parallel work can be scheduled through "jobtype=Normal" and saying "nodenumber=x" for the desired number of job slots.

scheduling to whole nodes

There is a need to schedule to whole nodes:

- MPI jobs don't want to share the CPU with other work
- shared memory communication is faster than across the network
- OpenMP or simple multi-threaded applications
- memory-intensive applications

Whole-node scheduling is demanding on the configuration of the local batch system. Some systems are more attuned to such use than others.

This leads to the next point: on a typical large, full cluster, the rollover of job slots happens in a more or less random pattern across the nodes. As nodes have multiple job slots, chances are your application is going to end up doing MPI across many different nodes, each of which is occupied with other work as well, and may be under different levels of stress. Since a typical MPI application is spreading the work evenly, a 'slow' node will hold up the entire run.

So there is a need, if only for optimization, to schedule work to as few physical nodes as possible; cluster managers who wish to optimize their site for MPI use should be well aware of these issues.

There is actually yet another reason to schedule whole nodes, and that is for jobs that want to exploit the possibilities of SMP (through, e.g., OpenMP) or jobs that are so high in their demand for main memory that they simply want all the memory to themselves, at the cost of leaving CPUs unused.

Granularity

As a more general system than whole-node scheduling, SMPGranularity can be used in the JDL;

```
NodeNumber=12
SMPGranularity=4
```

means that 12 cores in total, with a distribution of 4 cores per node, are to be allocated.

This new attribute requires some changes at least to the WMS and CREAM JDL. These have to be worked out further for the LCG-CE and for BLAH; the technicalities were just recently discussed within JRA1.

The possibilities for sites to address whole-node scheduling really depend on the local set-up and feature set of the batch system that is used. A site could dedicate a portion of the worker nodes to parallel jobs, and allow non-parallel work to use these nodes if there is no other work, but with a very short wall-time or with automatic migration to quickly free up the machines again.

Once the job lands on the worker node the next challenge is to distribute the program across the allocated nodes. The several options include: shared file system, passwordless ssh between nodes, automatically through the batch system. The mpi-start program should be able to pick the right method automatically. This reduces the burden on the user to design site-specific methods.

for administrators

- The current collection of mpi software for the worker nodes is outdated, and causes dependency issues with mpiexec. The packages need to be updated, build for different MPI implementations, tailored for use with different batch systems.
- Source RPMs should be provided to allow the site admin to make a local version if needed.
- Reviving and updating the SAM tests, basically checking how well the installation reflects the new recommendations. This both helps users and site admins.
- updating documentation
- updating the yaim functions, providing easy 'default' setup for simple clusters.

last but not least...

Documentation should be reviewed, updated, improved and augmented with examples, validation scripts, etc.

4 Summary

Summary of recommendations

- update packages, provide RPMs and source RPMs
- update yaim functions
- revitalize SAM tests
- SMPGranularity, with examples of integrating this with popular batch systems
- publishing flavour and version
- updating and improving documentation
- more structural support beyond the WG

References

[1] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006. Available from World Wide Web: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html`.

[2] The MPI Working Group. Wiki for MPI use in the EGEE grid. Available from World Wide Web: `http://www.grid.ie/mpi/wiki`.