Adaptive, Fault-tolerant MPI Applications with Dynamic Resource Allocation

Maciej Szpindler
m.szpindler@icm.edu.pl

University of Warsaw
Interdisciplinary Centre for Mathematical and Computational Modelling
http://www.icm.edu.pl

Agenda

- Introduction and motivation
- MPI and process management
- Dynamic resource allocation in MPI
- Application to MPI fault recovery
- Experimental results
- Summary

Intro and context

- MPI allows dynamic processes
- Do not precise interactions with underlying system components
  - i.e. queuing system, process binding and placement
- Resources allocation is done outside MPI runtime
  - Usually delegated to resource manager/queuing system
- Dynamic MPI processes with generic environment requires
  - Pre-allocation (spare cpus or nodes) - waste of resources
  - Process over-subscription - performance degradation
  - *but refer to yesterday morning talk on placement and oversubscription (HT)*
  - Sophisticated multi-application connectivity

Motivation

- Self-scheduling applications*
- Master-slave model
- Applications with variable computational load
- Fault tolerance and application recovery

* Hoefler, Torsten, et al. "Using Advanced MPI"
The MIT Press, 2014

MPI and PMI

- MPI (standard) do not define how to create new processes
- Process Management Interface* (PMI) – quasi standard
  - v1 and v2, PMI3 (?)
  - Abstract layer for inter-node process management
- Different implementations
  - Hydra – MPICH process manager (pm)
  - PMIx – OpenMPI effort (in development)
  - Slurm
  - Not necessarily compatible

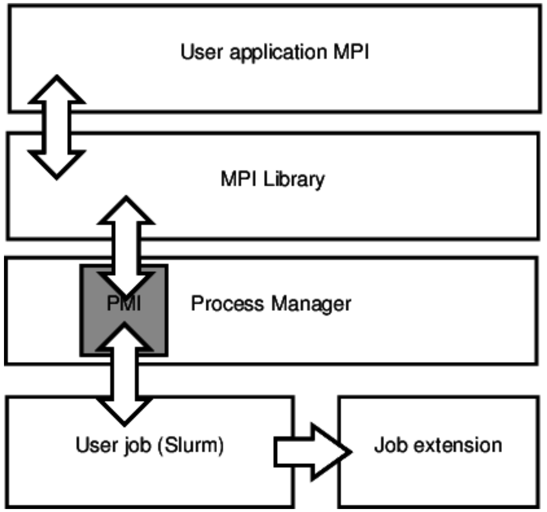
* Balaji, Pavan, et al. "PMI: A scalable parallel process-management interface for extreme-scale systems." *Recent Advances in the Message Passing Interface,* 2010

Dynamic Resource Allocation

- Idea based on the Slurm job resize method
  - Active allocation (already started job) is extended using additional, dependent allocation
  - Accesible from Slurm API
- Somehow inspired by dynalloc Slurm plugin (for hadoop??)
  - PMI calls Slurm API functions to extend allocation
  - It is a legal Slurm operation (no hacks)
- Eliminates need of pre-allocation or over-subscription of processes in case of dynamic MPI application

Dynamic process creation cascade

- MPI: MPI_Comm_spawn[_multiple]
  - API function, creates new processes
  - No control over exact startup parameters
  - **`Info`** argument theoretically passes additional requirements
- PMI: MPI – PMI interaction
  - With KVS pairs (key, value)
  - Parsed from MPI_Info structure
  - Comm_spawn is realized by PMI_Spawn
- Slurm:
  - Initializes slurm step and actually start process



2015, University of Warsaw
Interdisciplinary Centre for Mathematical and Computational Modelling

Resource allocation modes

- In reality, additional resources are not immediately available
  - Might be not practical for a range of applications
  - Blocking and non-blocking resource allocation if waiting is acceptable
  - Immediate allocation mode in the other case

Resource allocation modes cont.

- Blocking mode
  - Blocking mode returns control if resources are already allocated or given timeout reached (uses Slurm blocking API)
- Non-blocking mode
  - Most elegant option: MPI_Icomm_spawn + Wait (with Slurm callbacks)
  - Considered in the past by the MPI Forum as MPI extension*, but eventually dropped
  - Implementation is hard due to complicated MPI progress engine
  - Most practical: use helper thread for allocation, easy to implement
- Immediate mode
  - Returns extended allocation if resources are available immediately or raise an error
  - Uses Slurm allocation constraints

* Nonblocking Process Creation and Management Operations, MPI-Forum ticket https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/Async-proc-mgmt

Fault tolerance with MPI – ULFM

- User Level Failure Mitigation (ULFM)
  - proposal for the next MPI Standard version
- A set of primitives for application level
  - Failure detection
  - Failure notification
  - Error propagation
  - Communication recovery
- Implementations
  - OpenMPI (1.7, dedicated branch)
  - MPICH (3.2, almost complete)
- Common usage*
  - Detect – Revoke – Shrink - **Repair**

*Bland, Wesley, et al. "An evaluation of User-Level Failure Mitigation support in MPI." Computing, 2013

Failure detection - ULFM

- MPI supports error handling with:
  - MPI_ERRORS_ARE_FATAL error handler (default approach)
    - Immediately terminates all MPI processes
  - MPI_ERRORS_RETURN handler
    - Allows process local operation before termination
- ULFM follows second approach
  - Communication functions may raise:
    - MPI_PROC_FAILED error code in case of participating process failure
    - MPI_COMM_REVOKE in case of communicator being revoked
- Currently ULMF is extension
  - MPIX_[…] for MPICH, OMPI_[…] for OpenmMPI
- Running ULMF based applications
  - MPICH: mpiexec **-disable-auto-cleanup** …
  - OpenMPI (specific builds only): mpiexec **-am ft-enable-mpi** …

2015, University of Warsaw
Interdisciplinary Centre for Mathematical and Computational Modelling

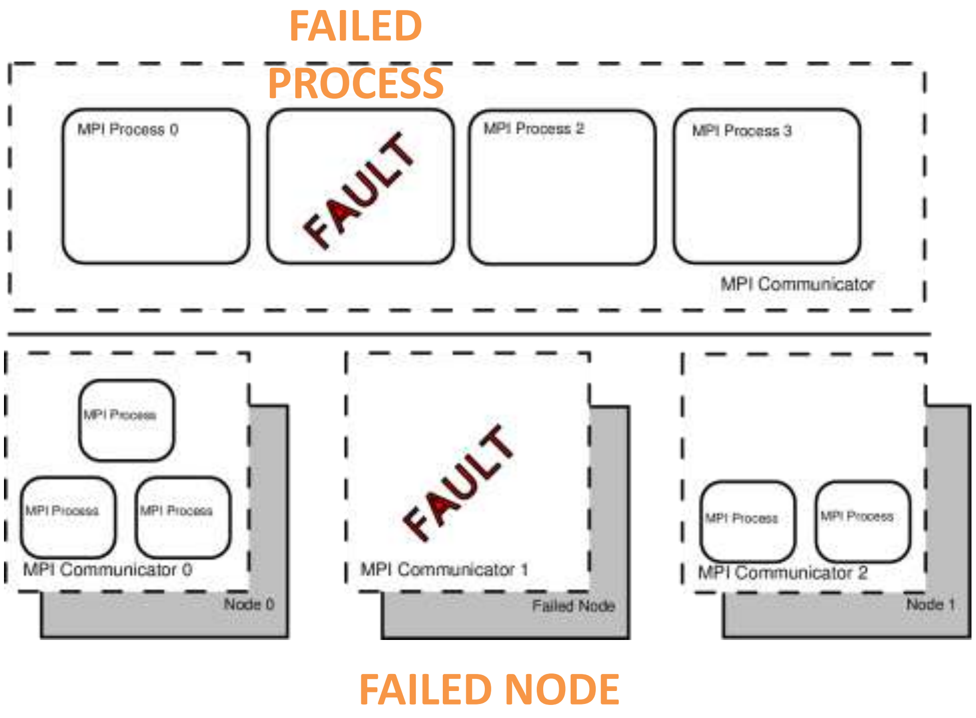
Communicator reconstruction

- Reconstruction (repair) in the ULMF model involves:
  - Excluding failed processes – shrink operation
  - Spawn new process(es) that replace failed ones (produce inter-communicators)
  - Merge inter-communicators
  - Optionally restore original rank order (if necessary)
- Application state recovery need additional effort
  - Recover process local state (at some point)
  - Possible approaches*: checkpoints, memory redundancy

* Ali, Md Mohsin, et al. "Application Level Fault Recovery: Using Fault-Tolerant Open MPI in a PDE Solver." *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014*

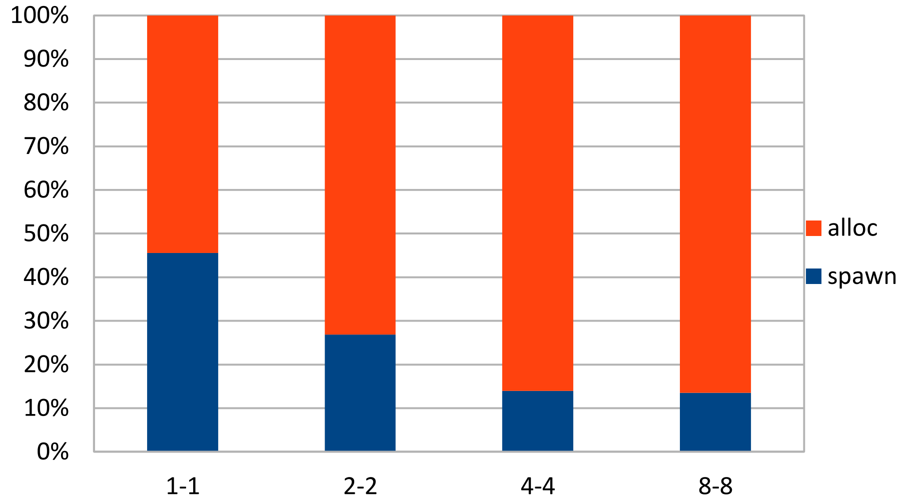
Common scenarios

- Single process failure (top)
  - One of the communicator's members fails
  - Process local memory vanishes
- Node failure (bottom)
  - All node's processes are lost
  - Node memory is lost
  - Node communicator(if used) is broken
  - Common case for MPI+OpenMP choice

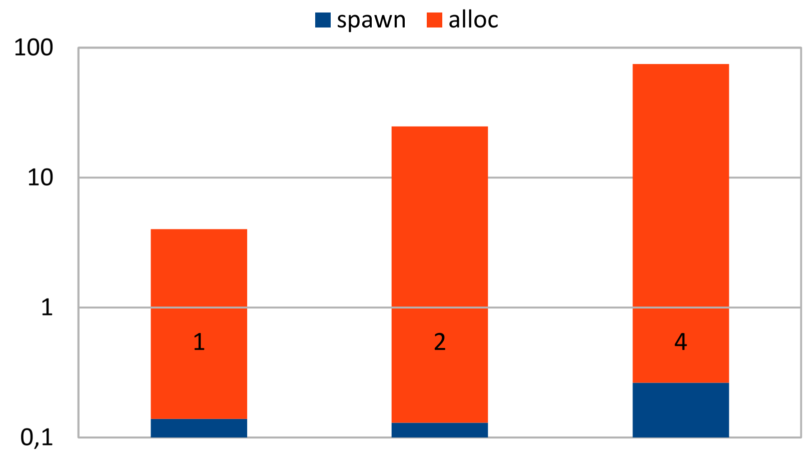


Results

- All experiments in immediate mode
- Synthetic mini-apps for exploring reconstruction with dynamic allocation
- Single process failure
  - Dynamic allocation on the local node (upper figure)
- Node failure
  - Dynamic allocation of the remote node (lower)
- Significant costs
  - MPI_Comm_Spawn is costly*
  - Order of magnitude slower is remote allocation



Relative cost of the spawn and allocate operations for increasing number of processes (N-M: number of parents and children)



Time cost in seconds of the spawn with remote node allocation. Note the logarithmic scale.

* Bland, Wesley, et al. "An evaluation of User-Level Failure Mitigation support in MPI." Computing , 2013

Results - technicalities

- Resource allocation time with Slurm highly dependent on the machine state (at least on the computer tested)
- Technical details
  - MPICH – easier PMI integration but lack of ULFM support for inter-communicators (v. 3.2a2)
  - OpenMPI – more ULFM supported but not in the mainline code, specific process manager, PMIx not integrated with distribution
  - Slurm memory management caused problems with hydra (MPICH)
- PMIx integration in progress

Summary

- Dynamic resource allocation for MPI with Slurm shown
- Basic integration with MPICH implemented
- Practical usage for ULFM based application recovery demonstrated
- Not machine specific (although Slurm specific)
- Areas for improvements indentified (e.g. remote allocation time)