CONTAINERS FOR HPC EXPERIENCE ON INTEL® XEON PHI™ PROCESSOR

Piotr Umiński, Intel

HPC Developers Conference 2017
Legal Disclaimers

Intel technologies features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel, the Intel logo, Intel. Experience What's Inside, the Intel. Experience What's Inside logo, Intel Xeon Phi, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

*Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.
Agenda

Overview of the container technology

Docker* containers on Intel® Xeon Phi™ processor

Using Singularity* for an MPI application on Intel® Xeon Phi™ processor

Performance results
About containers

Why do we need them?
• Flexibility
  • Multiple OS versions
    • Limited to the user space
  • Multiple versions of libraries, tools and config files
• Portability of the applications
  • Guarantee that all dependencies are installed on the target system
• Reproducibility
  • We can run application with identical dependences different systems

Can we use them for HPC?
• Yes, we can!
Containers abstract architecture

Shared OS kernel
- Works on bare metal
- Multiple user-space versions
- Shares the common kernel services
- Separated by OS mechanisms

Container run-time
- Controls container loading and execution
- Maintains illusion of separate filesystems per container
What is “micperf”? An HPC benchmarking tool for Intel® Xeon Phi™ processor
- Part of Intel® Xeon Phi™ Processor Software

Runs on a single node

Executes several benchmarks in a standard way:
- dgemm
- dgemm
- stream
- ...

Complex dependency lists
- Intel® Math Kernel Library (Intel® MKL)
- MPI
- Memkind library

Good candidate for a simple containerized application
Docker* containers

Leading container platform

- Maintained by Docker, Inc. - https://www.docker.com/

Execution controlled by a daemon

- Executed as a privileged user – potential security threat

Overlay file system

- Common layers are shared by multiple containers

Uses namespace and cgroup mechanisms to separate resources by containers
Micperf in the Docker* container

Dockerfile - entire container description, it is an instruction to:
  • Install base OS from the public repository.
  • Add Intel® MKL run-time from the archive.
  • Install memkind and micperf packages from RPMs.
  • Set Intel® MKL environment variables and run the application

Deployment
  • Docker* containers are stored in the repository
    • Either public or private repositories can be used
    • Proprietary format
    • Multiple layers to preserve the space
    • Automatically updated on run
  • Containers started manually or using orchestration frameworks
    • E.g. Kubernetes* (not used)
Micperf in the Docker* container

Security consideration

• For security reason Docker* blocks several system calls
• Micperf uses two blocked system calls to access MCDRAM:
  • mbind() and set_mempolicy()

• Solution 1: run container in the privileged mode.
• Solution 2: create and edit your own policy
Singularity* containers

Started as a project at Berkeley Lab - http://singularity.lbl.gov/
- Mission: create containers for HPC
- Maintained by the community
  - Commercialization of the product in progress

Integrated with MPI
- Open MPI, Intel® MPI

Containers executed as a regular user
- Do not use any demon

Each container has its own file system
- More space consumed (memory, disk)
- Easier to run and transfer
Micperf in the Singularity* container

Similar to Docker* but...
• Does not need a daemon owned by root
• Uses different container format
• Integrated with MPI

Container creation
• Create a container file
  • Regular file with an ext3 file system inside
• “Bootstrap” the container file
  • Copy from an existing Docker* container or install SW step-by-step
  • Step-by-step installation specified by shell commands
    • Can be done interactively by running shell commands inside the container

Deployment
• Container files shared via NFS
• Container started manually as a shell command
Docker*:  
**Adding Intel® Omni-Path Architecture support**

Install Intel® Omni-Path Software (Intel® OP Software) to the container:

- Install dependencies
- Install Intel® OP Software user-space
  - Intel® OP Driver from the bare metal kernel are used

Add middleware and benchmarks:

- Install Intel® OpenMP and Intel® MPI run-time.
- Install the distributable Intel® MKL library and benchmarks.

Exposing devices while running the container:

```
# docker run --privileged -it
--device=/dev/infiniband/rdma_cm --device=/dev/infiniband/uverbs0
--device=/dev/infiniband/ucm0 --device=/dev/infiniband/umad0
--device=/dev/hfi1_0 --ulimit memlock=-1
--network=host -v <volume> <container_name>
```
Docker*: running an MPI app

Intel® OmniPath Architecture™ (OPA) network
Docker*:  

**MPI via custom SSH port**

Add extra operations to Dockerfile

- Configure SSH access:
  - Set root password, allow root login.
  - Generate and set host key.

- Create a user account:
  - Add a user account.
  - Add user keys.
  - Configure ssh to use a non-default port.

- Inside the container run the ssh daemon on a non-standard port.

Dockerfile:
```
# Run sshd at port 9100
EXPOSE 9100
CMD /usr/sbin/sshd -p 9100 && /bin/sh
```

In the container:

- mpirun will use ssh on a non-standard port to access other containers.
Singularity*: running MPI apps

Singularity container started by mpirun

Compute node 1 (Mother Superior)

$mpirun <options> <container name>

SSH connection uses standard TCP port

Intel® OmniPatch Architecture™ (OPA) network

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Running MPI applications in Singularity*

Much simpler than in Docker*

• Similar installation of MPI and Intel® OP SW
• No need for SSH connection to the container

Singularity* container integrated with MPI

• A container can be started by mpirun from the head node
• Actual application command line must be specified as %runscript during container bootstrap

```
[head node]$ mpirun -hostfile hosts.txt -np 4 -ppn 1 singularity /containers/my_app.img
```
Bare metal vs containers performance: comparison for a single Intel® Xeon Phi™ processor

Near bare metal performance for both container types

Processor: Intel® Xeon Phi™ CPU 7250 68 cores @ 1.40GHz, memory mode: flat, quadrant, Red Hat* Enterprise Linux® Server release 7.3
Network: Intel(R) Omni-Path Architecture, SW build 10.3.1.0.22, Intel® MPI Library for Linux® OS, Version 2017 Update 1 Build 20161016, Intel® MKL version 2017.1.132
Results: average of 10 executions of each case

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.
Bare metal vs containers performance: comparison for multi-node

Near bare metal performance for both container types

- 4 Intel® Xeon Phi™ processors used
- Intel® OPA + Intel® MPI used for communication

Processor: Intel® Xeon Phi™ CPU 7250 68 cores @ 1.40GHz, memory mode: flat, quadrant, Red Hat® Enterprise Linux® Server release 7.3
Network: Intel(R) Omni-Path Architecture, SW build 10.3.1.0.22, Intel® MPI Library for Linux® OS, Version 2017 Update 1 Build 20161016, Intel® MKL version 2017.1.132

Results: average of 10 executions of each case

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit [http://www.intel.com/performance](http://www.intel.com/performance).
Conclusions

Containers can be used to make HPC applications portable and maintain performance

- Straightforward support for MPI in Singularity*
- MPI can be used with Docker* containers as well
- Performance of tested benchmarks on Intel® Xeon Phi™ processors is extremely close to bare metal

containers = near bare metal performance + flexibility
Resources

“Running Docker* Containers on Intel® Xeon Phi™ Processors”, White Paper, Intel Corporation 2017

“Building Containers for Intel® Omni-Path Fabrics using Docker* and Singularity*”, Application Note, Intel Corporation 2017