CRAY

ADAC6

Appropriate Use of Containers in HPC

Agenda

- Why HPC containers
- Investigate different container runtimes for HPC
- Integration with batch systems
- Best practices for HPC
- HPC Application performance

Why HPC containers

- **Mobility of computing to both users and HPC centers**
  - Means to capture and distribute software and compute environments
  - Entire workflow can be contained for others to replicate no matter what version of Linux they are running (Kernel ABI compatibility – syscall's)
- **Reproducibility of results – may not mean same performance**
  - MPI-ABI, libraries, host drivers
- **Standard and simplified packaging**
  - Independent Software Vendors (ISV) codes
  - Delivered benchmarks
  - CRI standard and/or propriety formats
- **Legacy workload packaging and execution**

Container Runtime Environments

- **Enterprise, and HPC container environments**
  - opencontainers/runc
    - **236** contributors, 17 release. 3,613 commits
    - Standard image format (Open Container Initiative – OCI)
  - rtk/rtk
    - **197** contributors, 65 releases , 5,539 commits
    - Standard image format (appc Specification – Application Container Image, ACI)
  - NERSC Shifter (github)
    - **11** contributors, 5 releases, 1,712 commits
    - Propriety image format
  - LBL Singularity (syslabs.io)
    - **63** contributors, 19 releases, 4,503 commits
    - Propriety image format

As of 06/18/18

Integration with batch systems

- **Examples of Moab/Torque and Cray ALPS**
  - Application based utilities
  - Limited or no cgroup and kernel namespace support

① $ aprun -n *N* -b **runc** --bundle /tmp/cle.img run $(date +%Y%m%d%H%M)

② $ aprun -n *N* … -b **rkt** run \
    --stage1-name=coreos.com/rkt/stage1-fly:1.21.0 \
    registry-1.docker.io/library/cle:latest --net=host --exec=/bin/hostname

③ $ aprun -n *N* … -b **shifter** --image cle:latest /bin/hostname

④ $ aprun -n *N* … -b **singularity** exec /global/cle.img /bin/hostname

Container Runtime Specifics

- Container runtimes configuration and semantics
- Unique image management
- Runtime arguments are specific

Configuration & Runtime options

- **runc**
  - Embedded in the OCI image definition: `config.json`
- **rkt**
  - /etc/rkt, /usr/lib/rkt, user-defined
    - Repository authentication, data and image locations
    - Command line can override system configurations
- **Shifter**
  - System configuration (/etc/shifter)
    - Authentication, data and image locations
- **Singularity**
  - System configuration $SYSCONFDIR/singularity/singularity.conf
    - Authentication, data and image locations

runc

- **Create a root file system, created via Docker container**

```
$ docker export alpine | tar xvfC - ./rootfs
```

- **Create container configuration file *config.json***

```
$ runc spec
$ vi config.json     # modify the args, see OCI specification
```

- **Run the container**

```
$ runc --log /dev/stdout run $(date +%Y%m%d%H%M)
```

OCI specification: https://github.com/opencontainers/runtime-spec/blob/master/runtime.md

rkt

- **Pull an image locallydocker2aci**

```
$ docker2aci docker://jsparkscraycom/cle:6.0.2
```

- **Run the container**

```
$ qsub –I
salloc: Granted job allocation 88
$ rkt run --no-overlay=true --net=host \
   --stage1-name=coreos.com/rkt/stage1-fly:1.22.0 \
    registry-1.docker.io/jsparkscraycom/cle:6.0.2 \
   ${options} –inherit-env=true \
   --environment=LD_LIBRARY_PATH=$LD_LIBRARY_PATH \
   --exec=/lus/${USER}/a.out -- myargs
```

Shifter

- **Pull an image**

```
$ shifterimg pull jsparkscraycom/test:latest
```

- **Run the container**

```
$ shifter --image jsparkscraycom/test:latest \
/usr/bin/date
```

Singularity

- **Pull an image (if needed)**

```
$ singularity pull docker://jsparkscraycom/test:latest
```

- **Run the container**

```
$ singularity exec \
    docker://jsparkscraycom/test:latest \
    /usr/bin/date
```

Container Best Practices

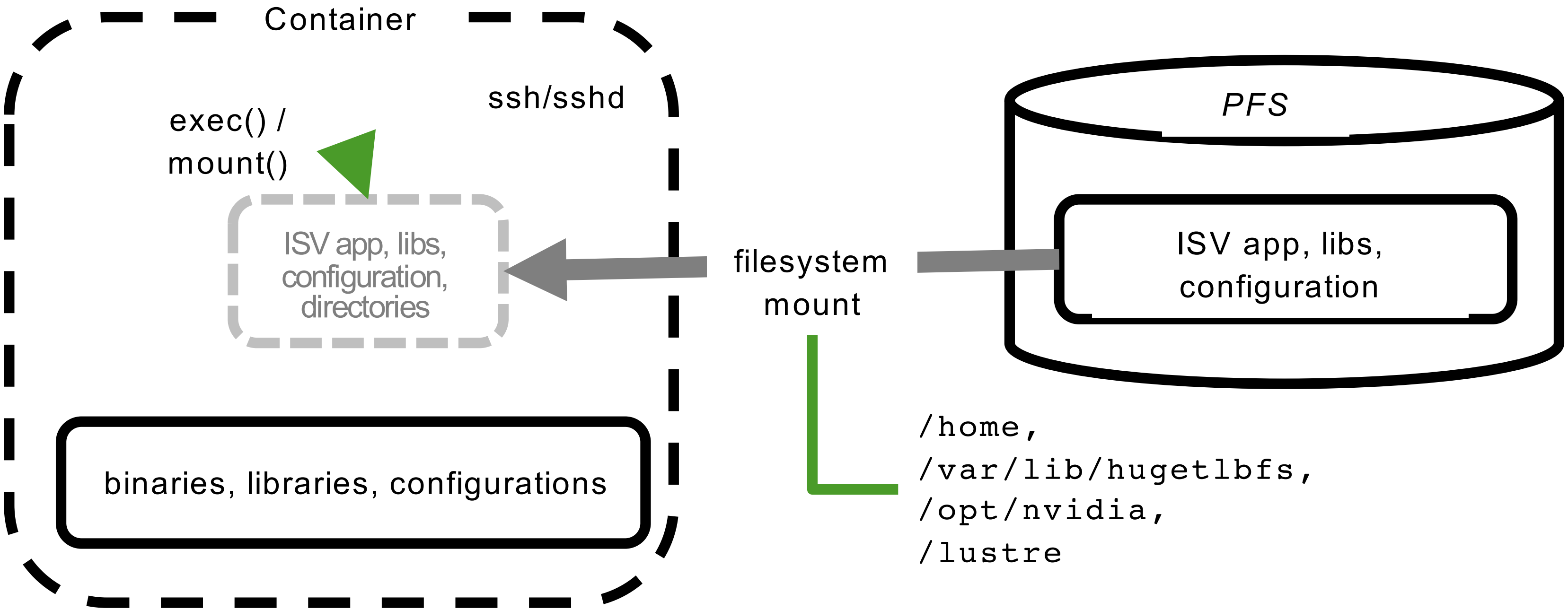
- **Portability**
  - Run anywhere vs. exploit host acceleration
  - Isolated contained environment, aka encapsulation – full vs. partial
- **Image standardization compliance**
  - OCI, ACI and custom (singularity)
- **HPC Optimizations**
  - Override container defaults via mounts and environment variables
    - Libraries and configurations
- **Framework independence**
  - No vendor/runtime lock in
- **Stripped frameworks**
  - opencontainers/runc, Charliecloud, rtk

Containerization Models

- **Enterprise case**
  - Standard runtimes (docker/rkt)
  - Fully encapsulated
    - Everything the application needs is in the container
  - Single application
  - Orchestration software
  - Standard image formats

- **HPC case**
  - HPC runtimes (Shifter/Singularity/...)
  - Partial encapsulation – limited namespaces
    - Access to host resources – networks, storage
    - blurs the line between container and host such that local directories can exist within the container.
  - Multiple applications
  - Batch system integration
  - Image format for scale (pfs)

HPC Partial Encapsulation

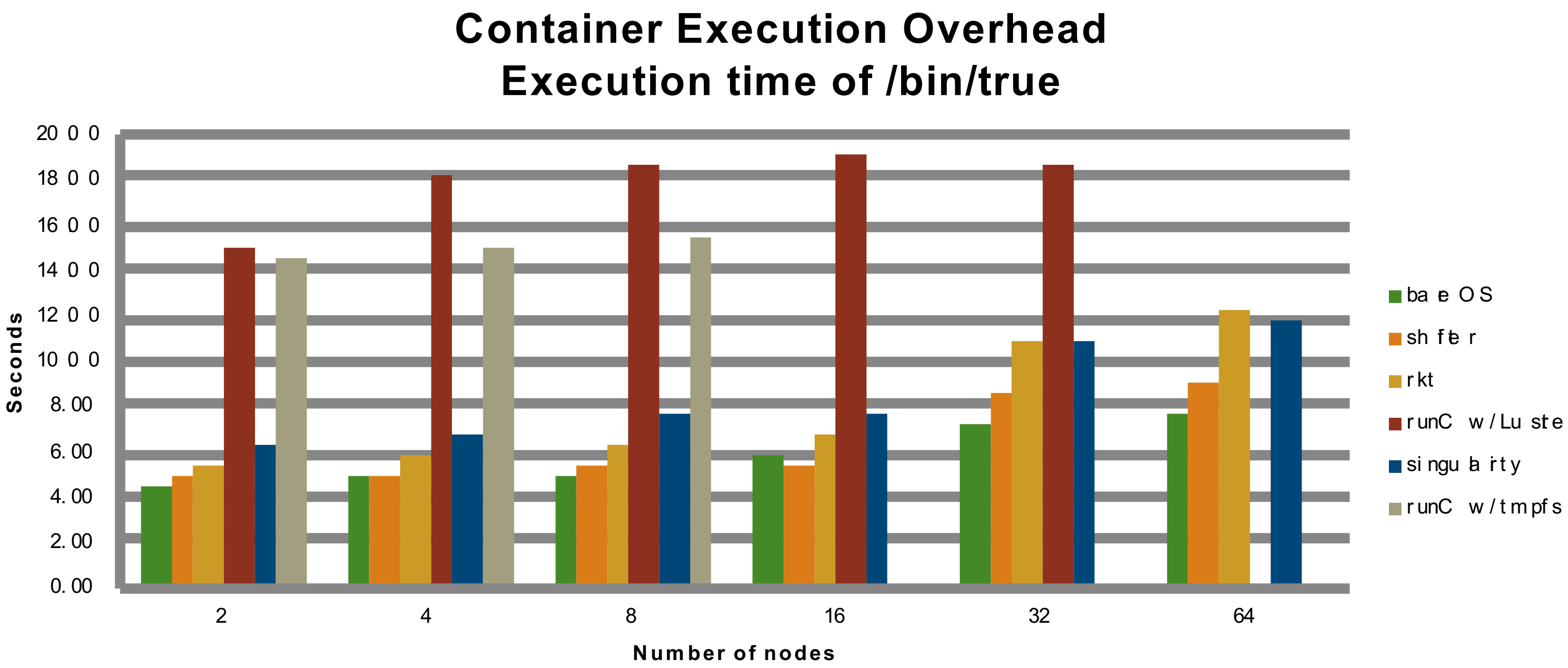


HPC Application performance

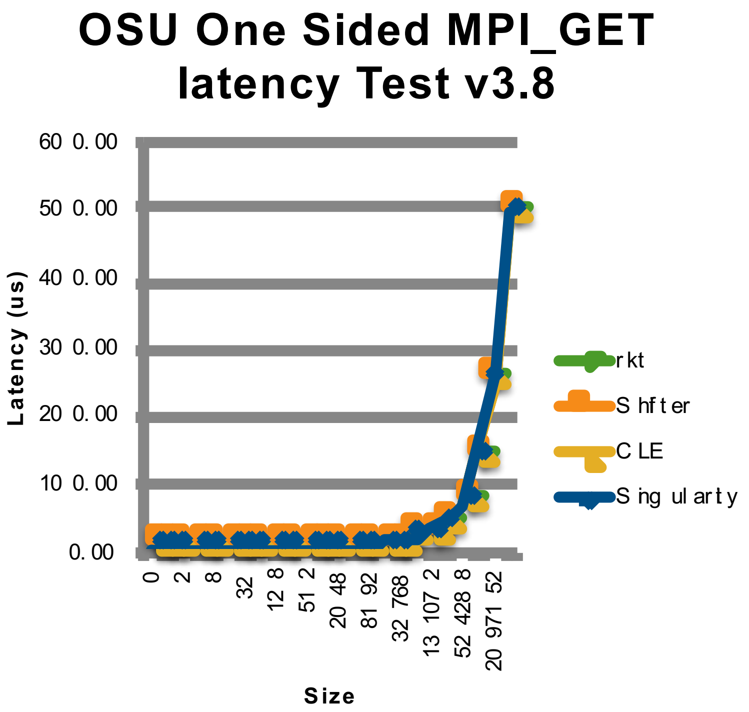
- **Launch times**
  - Time to setup and launch via container runtime
- **Application performance**
  - Hugepage optimization
  - Environment pass-through

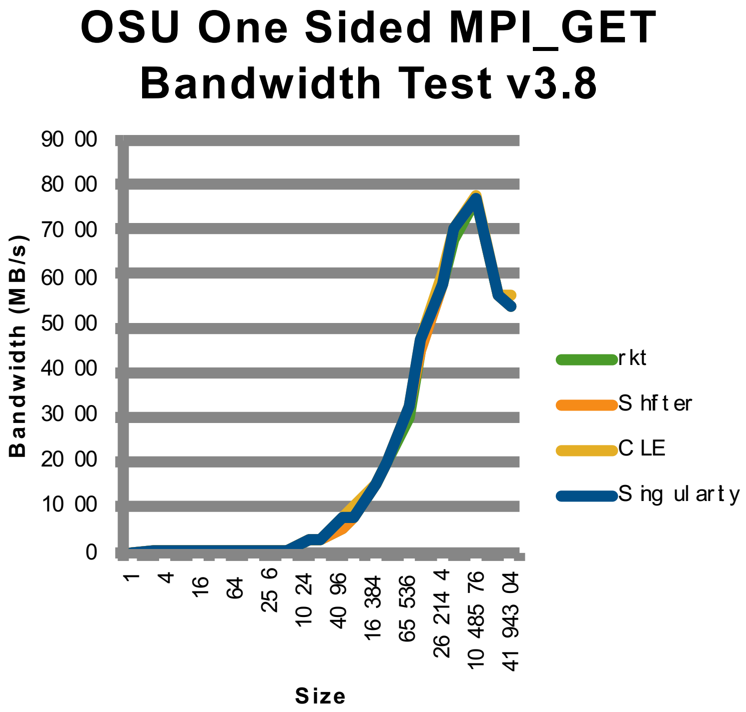
An Updated Performance Comparison of Virtual Machines and Linux Containers
Ref: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf

Launch Times

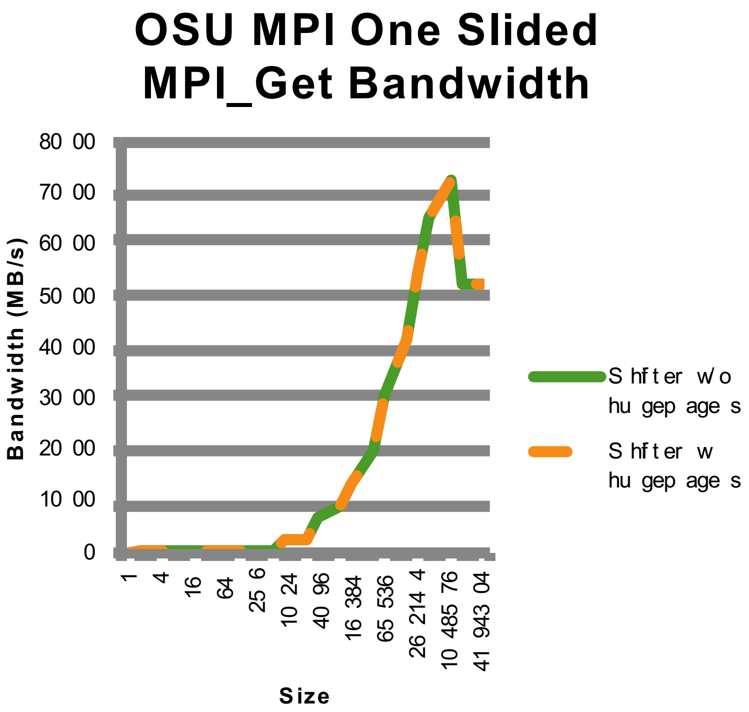


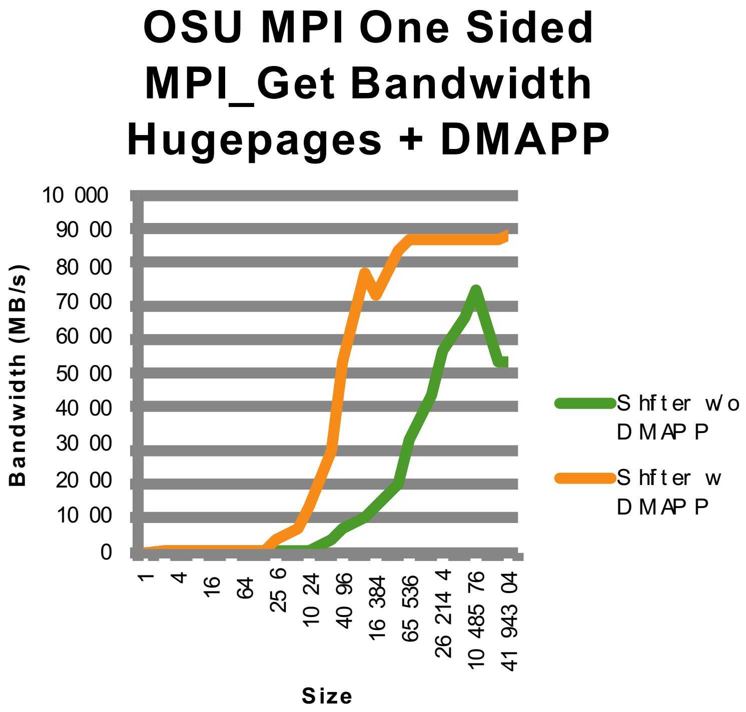
OSU Benchmarks



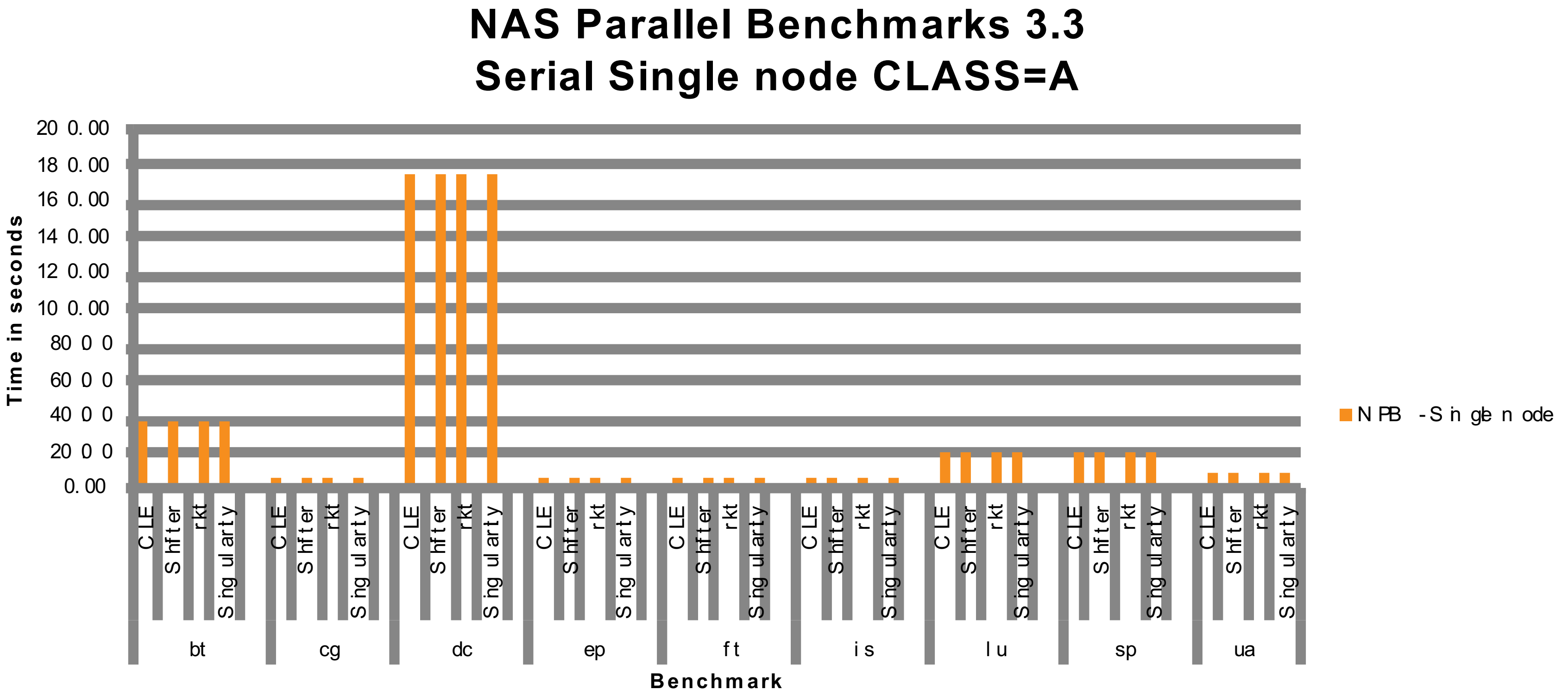


Hugepage / DMAPP

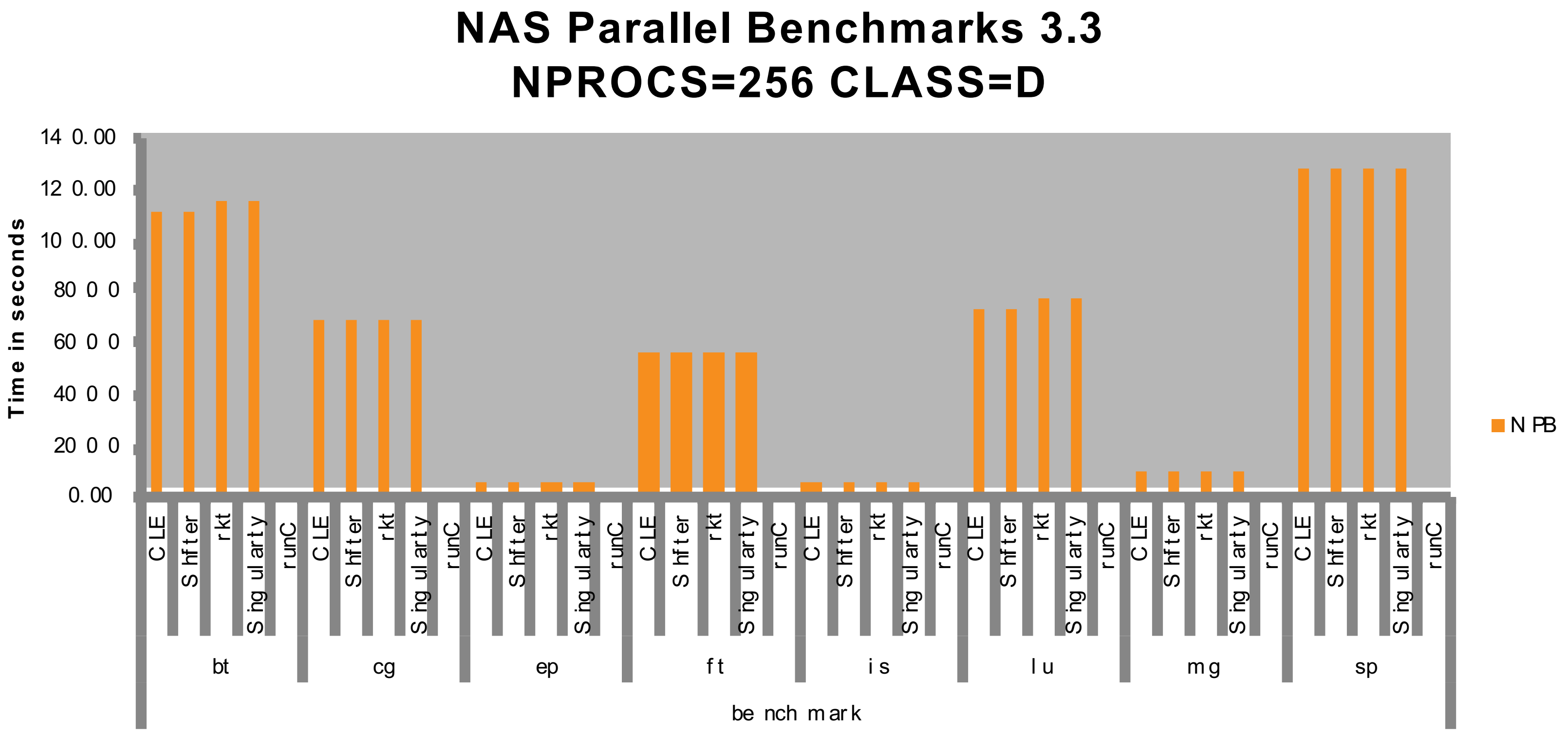




NPB Single node

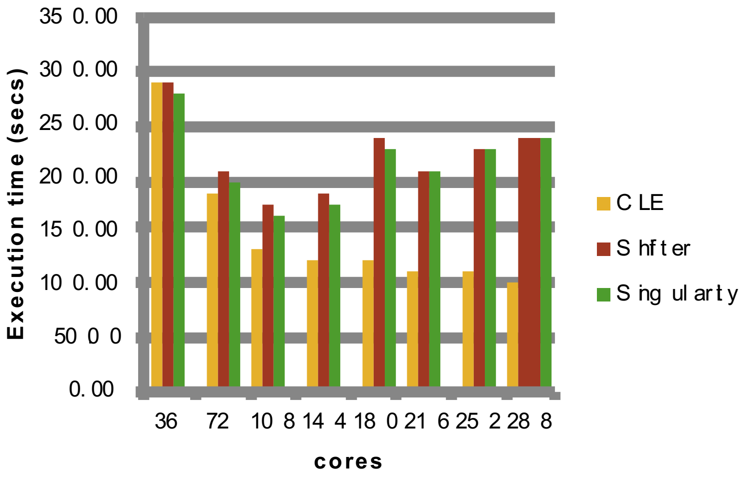


NPB Multi-node

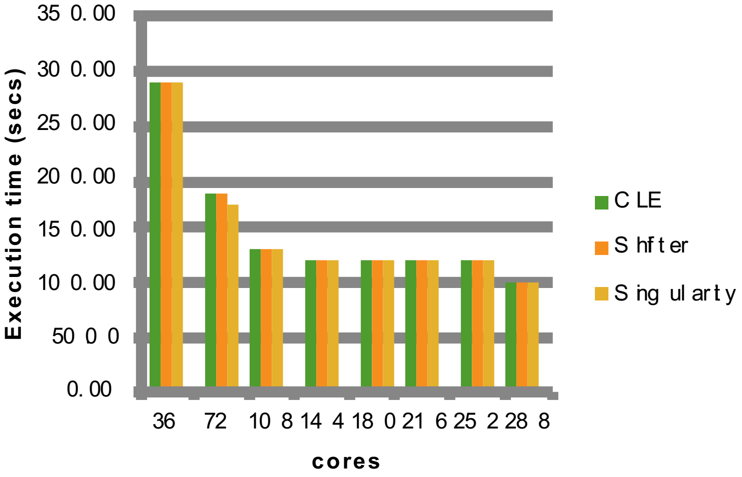


Quantum ESPRESSO – optimization

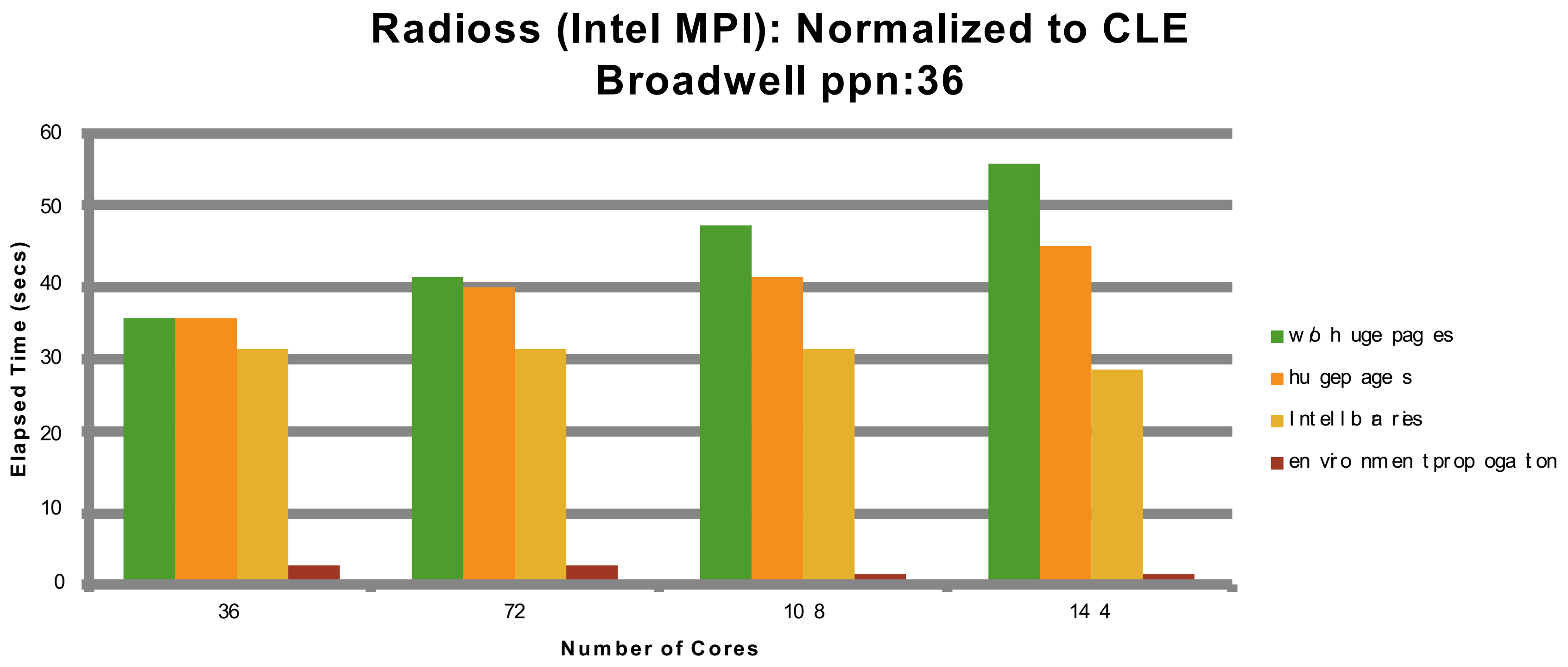
Execution time of QUANTUM ESPRESSO 6.0 / Broadwell none-hugepage



Execution time of QUANTUM ESPRESSO 6.0 / Broadwell hugepage



Radioss – optimization



Conclusions and Future Work

- **Deployment**
  - Enterprise frameworks required per-node image caches
  - Enterprise frameworks, as expected offer more runtime options
  - HPC frameworks enforce strict user security
- **Performance**
  - Native application performance can be achieved, requires host-level access to resources (network, file system)
  - Host environment pass-through
  - Launch time dependent on container infrastructure and image size
- **Future work**
  - Scaling investigation of open container frameworks
    - Shared image storage – faster startup times
  - Abstraction layer standardizing on applications packaging, deployed and run in isolation – OCI.

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.