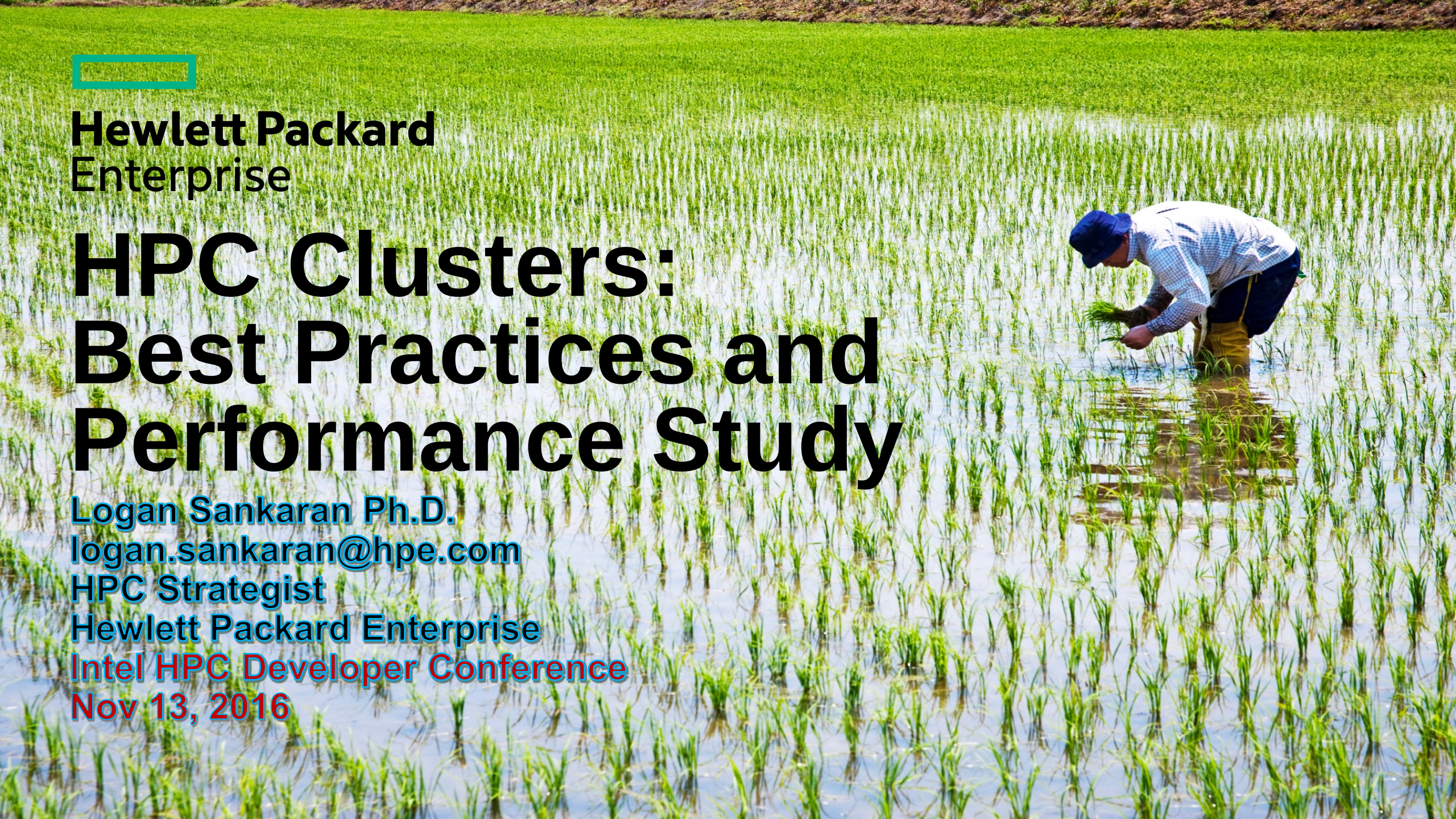


Hewlett Packard Enterprise

HPC Clusters: Best Practices and Performance Study

Logan Sankaran Ph.D.
logan.sankaran@hpe.com
HPC Strategist
Hewlett Packard Enterprise

Intel HPC Developer Conference
Nov 13, 2016

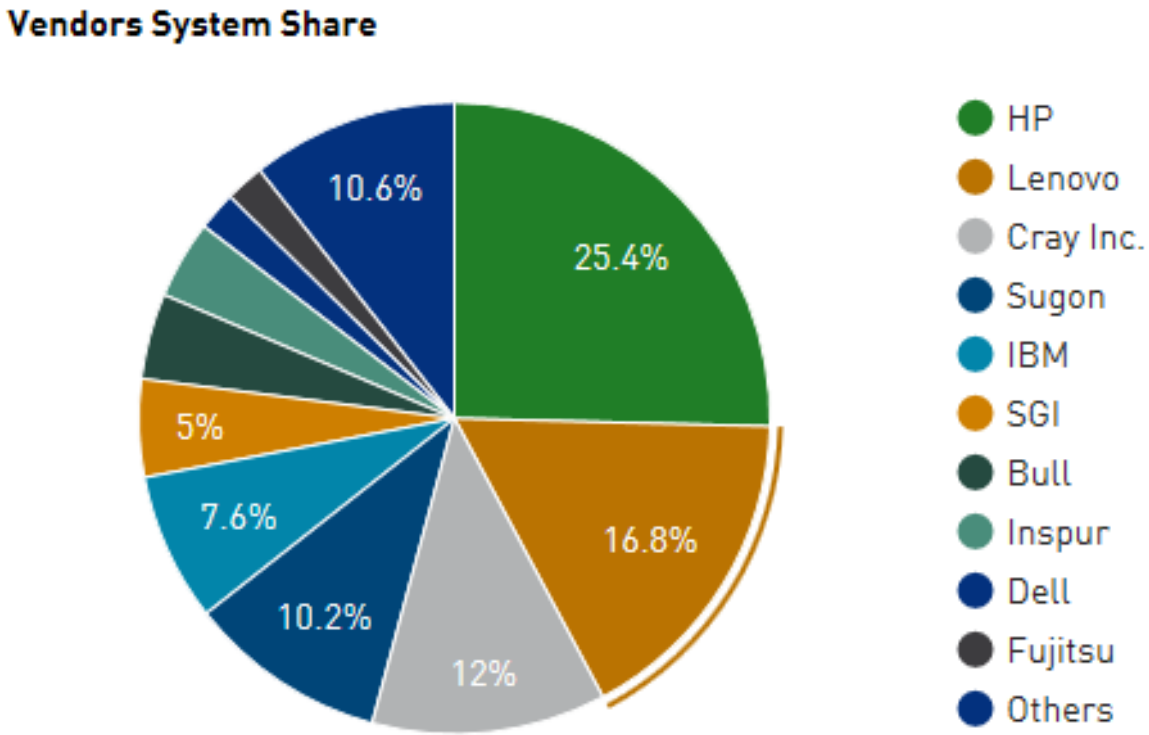
Agenda

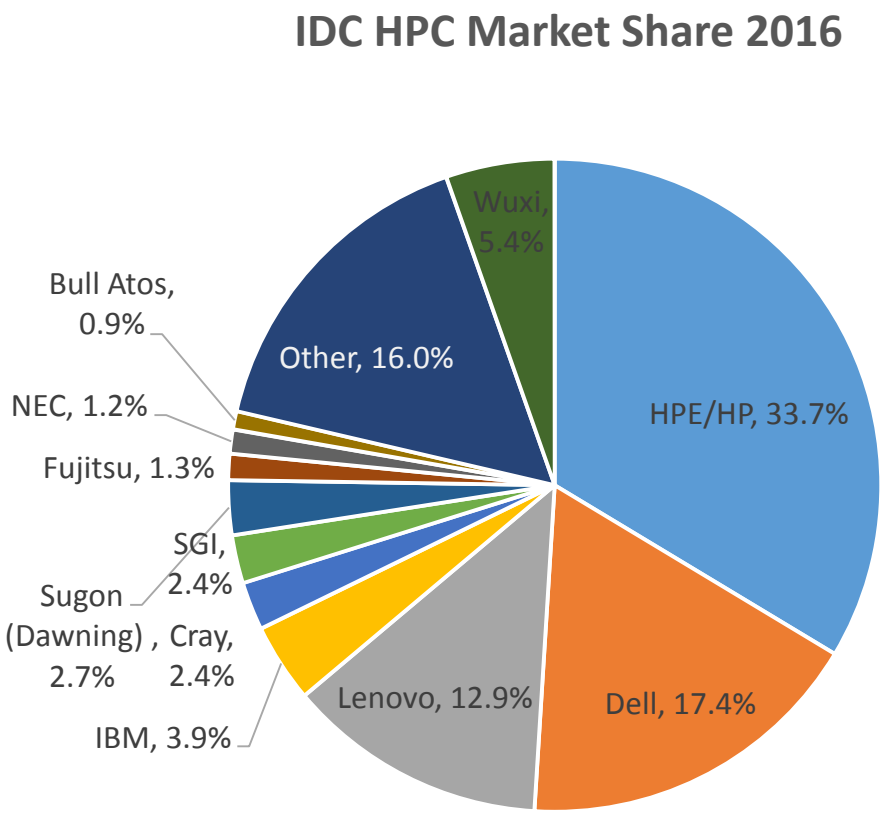
- HPC at HPE
- System Configuration and Tuning
- Best Practices for Building Applications
- Intel Xeon Processors
- Efficient Methods in Executing Applications
- Tools and Techniques for Boosting Performance
- Application Performance Highlights
- Conclusions

HPC at HPE

HPE's HPC Market and Share

Top500 List (ISC2016, June 2016)





Hewlett Packard Enterprise

System Configuration and Tuning

Typical BIOS Settings: Processor Options

– Hyperthreading Options Disabled	:	Better scaling for HPC workloads
– Processor Core Disable 0	:	Enables all available cores
– Intel Turbo Boost Technology Enabled	:	Increases clock frequency (increase affected by factors).
– ACPI SLIT Preferences Enabled	:	OS can improve performance by efficient allocation of resources among processor, memory and I/O subsystems.
– QPI Snoop Configuration Home/Early/COD	:	Experiment and set the right configuration for your workload. **Home**: High Memory Bandwidth for average NUMA workloads. **COD**: Cluster On Die, Increased Memory Bandwidth for optimized and aggressive NUMA workloads. **Early**: Decreases latency but may also decrease memory bandwidth compared to other two modes.

Typical BIOS Settings: Power Settings and Management

– HPE Power Profile should be set to Maximum Performance to get best performance (idle and average power will increase significantly).

– Custom Power Profile will reduce idle and average power at the expense of 1-2% performance reduction.

– To get highest Turbo clock speeds (when partial cores are used), use Power Savings Settings.

BIOS Configuration	Maximum Performance	Power Savings
HP Power Profile	Maximum Performance	Custom
HP Power Regulator	HP Static High Performance Mode	OS Control Mode
Intel Minimum Processor Idle Power State	No C-States	C6 State
Intel Minimum Processor Idle Power Package State	No Package State	Package C6 State

– For Custom Power Profile, you will have to set the following additional settings:

BIOS Configuration	Maximum Performance	Power Savings
Energy/Performance Bias	Maximum Performance	Maximum Performance
Dynamic Power Savings mode Response	Fast	Fast
Collaborative Power Control	Disabled	Enabled

Best Practices for Building Applications

Building Applications: Intel Compiler Flags

-O2	enable optimizations (= -O, Default)
-O1	optimize for maximum speed, but disable some optimizations which increases code size for small speed benefit
-O3	enable –O2 plus more aggressive optimizations that may or may not improve performance for all programs.
-fast	enable –O3 –ipo –static
-xHOST	optimize code based on the native node used for compilation
-xAVX	enable advanced vector instructions set (for Ivy Bridge performance)
-xCORE-AVX2	enable advanced vector instructions set 2 (key Haswell/Broadwell performance)
-xMIC-AVX512	enable advanced vector instructions set 512 (for future KNL/SkyLake based systems)
-mp	maintain floating-point precision (disables some optimizations)
-parallel	enable the auto parallelizer to generate multi-threaded code
-**openmp**	generate multi-threaded parallel code based on OpenMP directives
-ftz	enable/disable flushing denormalized results to zero
-opt-streaming-stores [always **auto** never]	generates streaming stores
-mcmodel=[small medium large]	controls the code and data memory allocation
-fp-model=[fast precise source strict]	controls floating point model variation
-mkl=[parallel sequential cluster]	link to Intel MKL Lib. to build optimized code.

Building Applications: Compiling Thread Parallel Codes

pgf90 –mp –O3 –Mextend –Mcache_align –k8-64 *ftn.f*

pathf90 –mp –O3 –extend_source –march=opteron *ftn.f*

ifort –openmp –O3 -132 –i_dynamic –ftz –IPF_fma *ftn.f*

pgcc –mp –O3 –Mcache_align –k8-64 *code.c*

opencc –mp –O3 –march=opteron *code.c*

icc –openmp –O3 –i_dynamic –ftz –IPF_fma *code.c*

Combination Flags

Intel:	-fast	=>	-O3 –ipo –static
PGI:	-fast	=>	-O2 –Munroll –Mnoframe
Open64:	-Ofast	=>	-O3 -ipa -OPT:Ofast -fno-math-errno

Notes:

- Must compile and link with –mp / –openmp
- Aggressive optimizations may compromise accuracy

Building Applications: Compiling MPI based Codes

mpicc	C compiler wrapper to build parallel code
mpiCC	C++ compiler wrapper
mpif77	Fortran77 compiler wrapper
mpif90	Fortran90 compiler wrapper
mpirun	command to launch mpi parallel job

Environment Variables to specify the Compilers to use:

```
export I_MPI_CC=icc
export I_MPI_CXX=icpc
export I_MPI_F90=ifort
export I_MPI_F77=ifort
```

Building Applications: Compiling MPI based Codes (Contd…)

mpif90 –O3 –Mextend –Mcache_align –k8-64 *ftn.f*
mpif90 –O3 –extend_source –march=opteron *ftn.f*
mpif90 –O2 -xHOST –fp-model strict -openmp *ftn.f*
mpicc –O3 –Mcache_align –k8-64 *code.c*
mpicc –O3 –march=opteron *code.c*
mpicc –O3 -xAVX2 -openmp –ftz –IPF_fma *code.c*

Compilers and Interface chosen depend on:

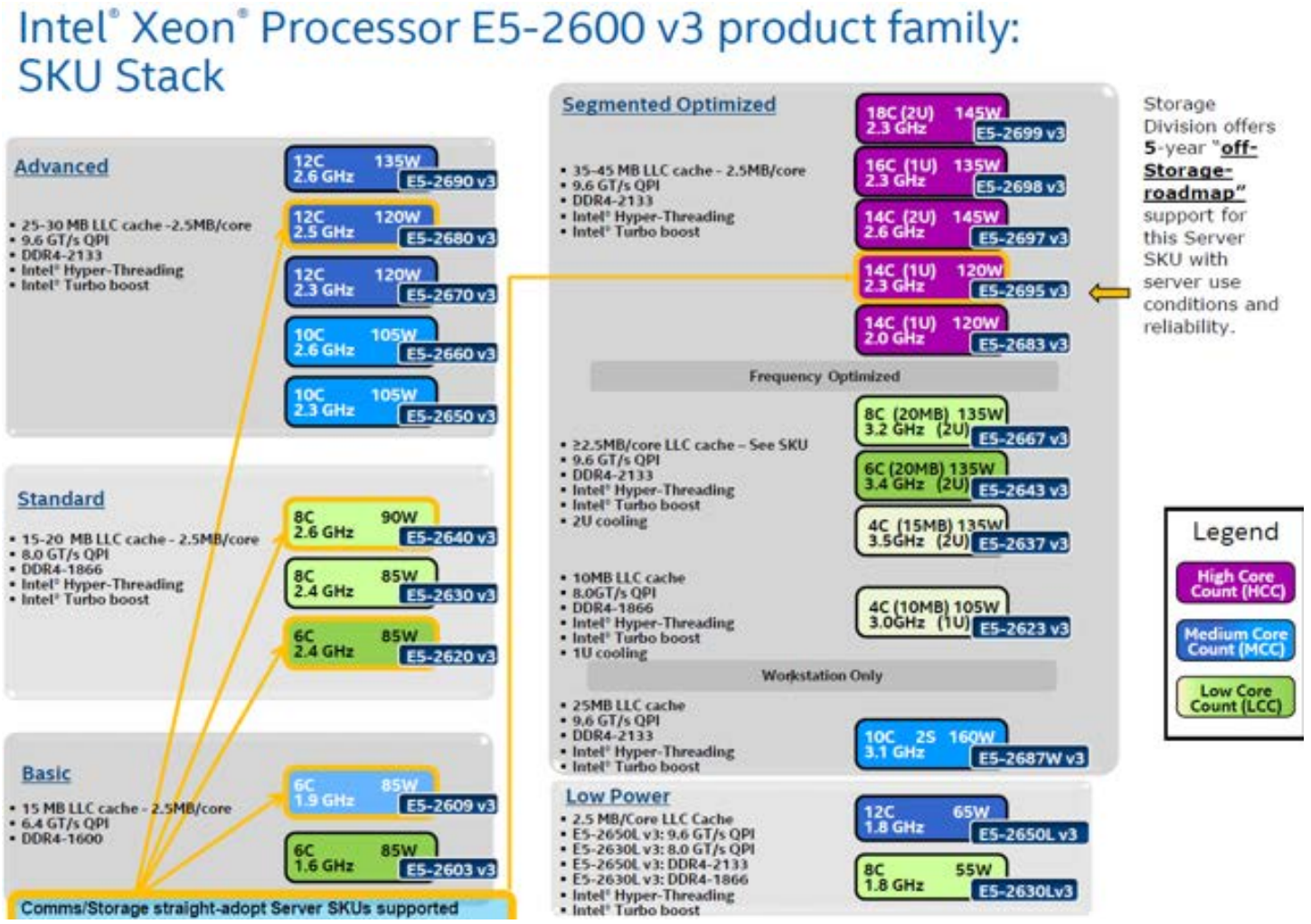
what is defined in your PATH variable

what are defined by (for Intel MPI):

- I_MPI_CC, I_MPI_CXX
- I_MPI_F77, I_MPI_F90

Intel Xeon Processor

Intel Xeon Processors: Turbo, AVX and more



Complete Specifications at:
http://www.intel.com/content/www/us/en/processors/xeon/xeon-e5-v3-spec-update.html

Intel Xeon Processors: Turbo, AVX and more (Contd …)

Non-AVX Turbo Boost 2.0 Frequency Bin upside by SKU†

Intel® Xeon® Processor E5-2600 v3 product family: Advanced, Standard, & Basic SKUs

Processor SKU	Base Frequency (GHz)	Cores	Cache (MB)	Maximum Frequency in GHz (+ x00 MHz over base frequency)															
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
E5-2690 v3	2.6	12	30	+9	+9	+7	+6	+5	+5	+5	+5	+5	+5	+5	+5	n/a	n/a	n/a	n/a	n/a	n/a
E5-2680 v3	2.5	12	30	+8	+8	+6	+5	+4	+4	+4	+4	+4	+4	+4	+4	n/a	n/a	n/a	n/a	n/a	n/a
E5-2670 v3	2.3	12	30	+8	+8	+6	+5	+4	+3	+3	+3	+3	+3	+3	+3	n/a	n/a	n/a	n/a	n/a	n/a
E5-2660 v3	2.6	10	25	+7	+7	+5	+4	+3	+3	+3	+3	+3	+3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2650 v3	2.3	10	25	+7	+7	+5	+4	+3	+3	+3	+3	+3	+3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2640 v3	2.6	8	20	+8	+8	+6	+5	+4	+3	+2	+2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2630 v3	2.4	8	20	+8	+8	+6	+5	+4	+3	+2	+2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2620 v3	2.4	6	15	+8	+8	+5	+4	+3	+2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2609 v3	1.9	6	15	Intel® Turbo Boost Technology not supported																	
E5-2603 v3	1.6	6	15	Intel® Turbo Boost Technology not supported																	

Complete Specifications at:
http://www.intel.com/content/www/us/en/processors/xeon/xeon-e5-v3-spec-update.html

Intel Xeon Processors: Turbo, AVX and more (Contd…)

AVX Turbo Boost 2.0 Frequency Bin upside by SKU†

Intel® Xeon® Processor E5-2600 v3 product family: Advanced, Standard, & Basic SKUs

Processor SKU	AVX Base Freq (GHz)	Cores	Cache (MB)	Maximum Frequency in GHz (+ x00 MHz over base frequency)																	
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
E5-2690 v3	2.3	12	30	+9	+9	+7	+7	+7	+7	+7	+7	+7	+7	+7	+7	n/a	n/a	n/a	n/a	n/a	n/a
E5-2680 v3	2.1	12	30	+10	+10	+8	+7	+7	+7	+7	+7	+7	+7	+7	+7	n/a	n/a	n/a	n/a	n/a	n/a
E5-2670 v3	2.0	12	30	+9	+9	+7	+6	+6	+6	+6	+6	+6	+6	+6	+6	n/a	n/a	n/a	n/a	n/a	n/a
E5-2660 v3	2.2	10	25	+9	+9	+7	+7	+7	+7	+7	+7	+7	+7	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2650 v3	2.0	10	25	+8	+8	+6	+6	+6	+6	+6	+6	+6	+6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2640 v3	2.2	8	20	+12	+12	+10	+9	+8	+7	+6	+6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2630 v3	2.1	8	20	+11	+11	+9	+8	+7	+6	+5	+5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2620 v3	2.1	6	15	+11	+11	+8	+7	+6	+5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
E5-2609 v3	1.9	6	15	Intel® Turbo Boost Technology not supported																	
E5-2603 v3*	1.3	6	15	Intel® Turbo Boost Technology not supported																	

Complete Specifications at:
http://www.intel.com/content/www/us/en/processors/xeon/xeon-e5-v3-spec-update.html

Intel Xeon Processors: Turbo, AVX and more (Contd …)

Intel publishes 4 different reference frequencies for every Xeon Processor:

1. Base Frequency 2. Non-AVX Turbo 3. AVX Base Frequency 4. AVX Turbo

- Turbo clock for a given model can vary as much as 5% from one processor to another
- Four possible scenarios exist:
  - Turbo=OFF and AVX=NO => Clock is set to Base frequency
  - Turbo=ON and AVX=NO => Clock range will be from Base to Non-AVX Turbo
  - Turbo=OFF and AVX=YES => Clock range will be from AVX Base to Base Frequency
  - Turbo=ON and AVX=YES => Clock range will be from AVX Base to AVX Turbo

Efficient Methods in Executing Applications

Running Parallel Programs in a Cluster: Intel MPI

– Environments in General
  – export PATH
  – export LD_LIBRARY_PATH
  – export MPI_ROOT
  – export I_MPI_FABRICS= shm:dapl
  – export I_MPI_DAPL_PROVIDER=ofa-v2-mlx5_0-1u
  – export NPROCS=256
  – export PPN=16
  – export I_MPI_PIN_PROCESSOR_LIST 0-15
  – export OMP_NUM_THREADS=2
  – export KMP_STACKSIZE=400M
  – export KMP_SCHEDULE= static,balanced
– Example Command using Intel MPI
  – time mpirun -np $NPROCS -hostfile ./hosts -genvall –ppn $PPN –genv I_MPI_PIN_DOMAIN=omp ./myprogram.exe

Profiling a Parallel Program: Intel MPI

– Using Intel MPS (MPI Performance Snapshot)
  – Set all env variables to run Intel MPI based application
  – Source the following additionally:
    source /opt/intel/16.0/itac/9.1.2.024/intel64/bin/mpsvars.sh –papi | vtune
  – Run your application as: mpirun –mps -np $NPROCS -hostfile ./hosts ….
  – Two files app_stat_xxx.txt and stats_xxx.txt will be available at the end of the job.
  – Analyze the these *.txt using mps tool
  – Sample data you can gather from:
    – Computation Time: 174.54 sec 51.93%
    – MPI Time: 161.58 sec 48.07%
    – MPI Imbalance: 147.27 sec 43.81%
    – OpenMP Time: 155.79 sec 46.35%
    – I/O wait time: 576.47 sec (0.08 %)

– Using Intel MPI built-in Profiling Capabilities
  – Native mode: mpirun -env I_MPI_STATS 1-4 -env I_MPI_STATS_FILE native_1to4.txt …
  – IPM mode: mpirun -env I_MPI_STATS ipm -env I_MPI_STATS_FILE ipm_full.txt

Tools and Techniques for Boosting Performance

Tools, Techniques and Commands

– Check Linux pseudo files and confirm the system details

– cat /proc/cpuinfo	>> provides processor details (Intel's tool cpuinfo.x)
– cat /proc/meminfo	>> shows the memory details
– /usr/ sbin /ibstat	>> shows the Interconnect IB fabric details
– /sbin/sysctl –a	>> shows details of system (kernel, file system etc.)
– /usr/bin/lscpu	>> shows cpu details including cache sizes
– /usr/bin/lstopo	>> shows the hardware topology
– /bin/uname -a	>> shows the system information
– /bin/rpm –qa	>> shows the list of installed products including versions
– cat /etc/redhat-release	>> shows the redhat version
– /usr/sbin/dmidecode	>> shows system hardware and other details (need to be root)
– /bin/ dmesg	>> shows system boot-up messages
– /usr/bin/numactl	>> checks or sets NUMA policy for processes or shared memory
– /usr/bin/taskset	>> shows cores and memory of numa nodes of a system

Top10 Practical Tips for Boosting Performance

– Check the system details thoroughly (Never assume !)

– Choose a compiler and MPI to build your application (All are not same !)

– Start with some basic compiler flags and try additional flags one at a time (Optimization is incremental !)

– Use the built-in libraries and tools to save time and improve performance (Libs., Tools are your friends !)

– Change compiler and MPI if your code fails to compile or run correctly (Trying to fix things is futile !)

– Test your application at every level to arrive at an optimized code (Remember the 80-20 rule !)

– Customize your runtime environment to achieve desired goals (process parallel, hybrid run etc.)

– Always place and bind the processes and threads appropriately (Life saver !)

– Gather, check and correct your runtime environment (what you get may not be what you want !)

– Profile and adjust optimization and runtime environments accordingly (Exercise caution !)

Application Performance Highlights

Application: High Performance Linpack (HPL)

Description:

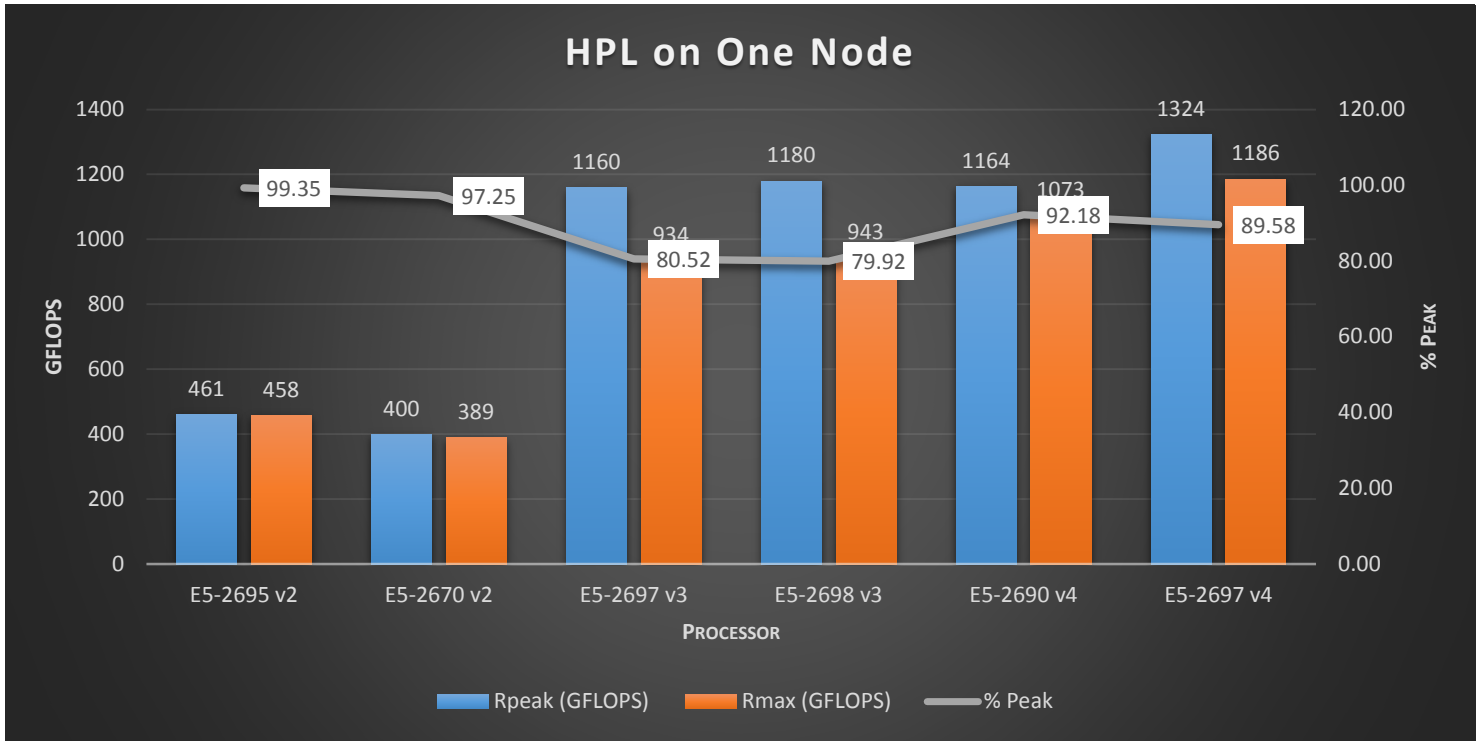
Benchmark to measure floating point rates (and times) by solving a random dense linear system of equations in double-precision.

- Originally developed by Jack Dongarra at Univ. of Tennessee.
- Used Intel optimized HPL binary for this study.
- Ran the code in hybrid mode, one MPI process per processor and each process launched threads equal to no. of cores on the processor.
- Used explicit placing and binding of threads.
- Attempted various choices of array sizes and other parameters to identify the best performance.
- The code provides a self-check to validate the results.

Additional details at: http://icl.eecs.utk.edu/hpl/

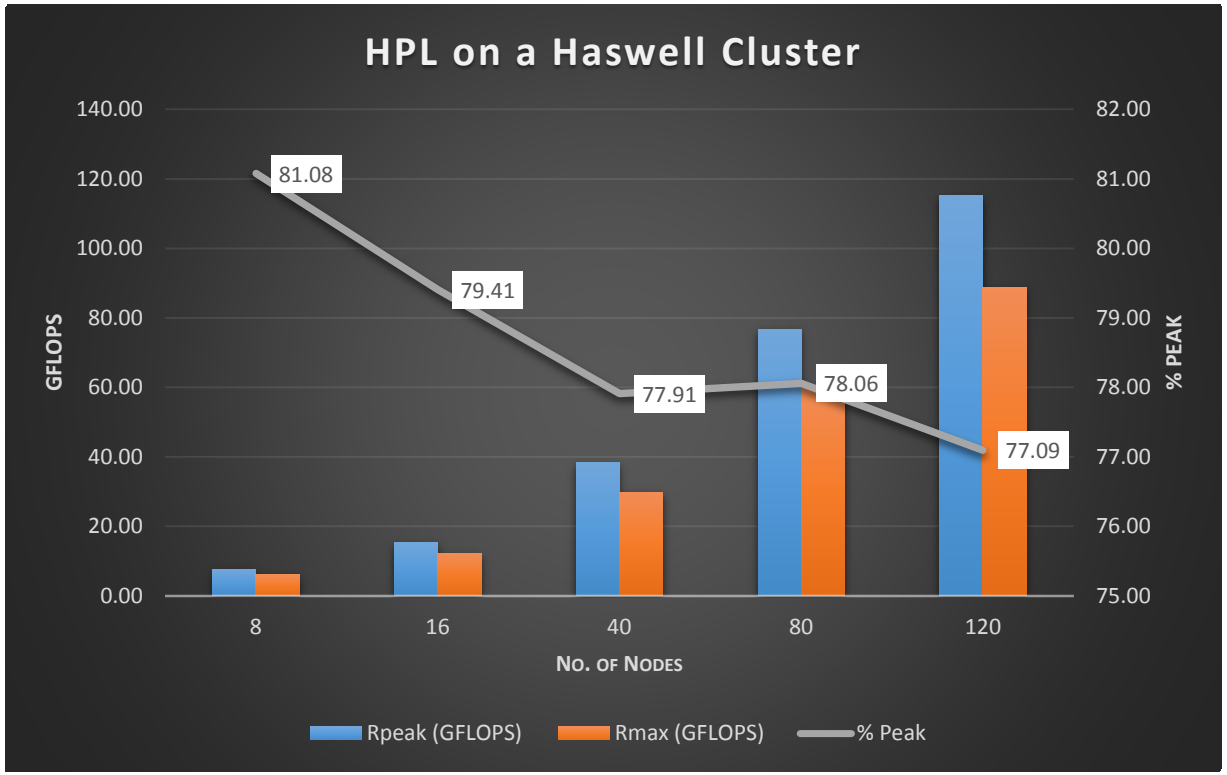
HPL Performance from a Single Node

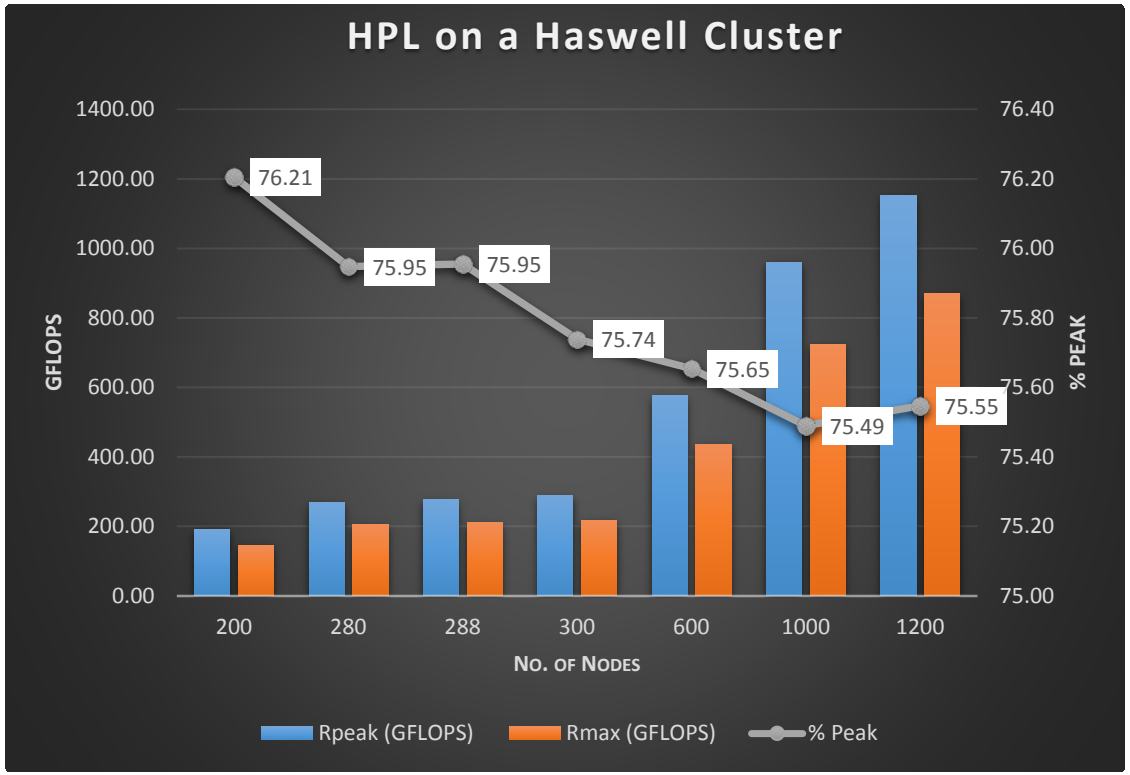
Proc.Type	Processor	Clock (GHz)	# cores/proc	#cores/node	TDP (Watts)	L3 Cache (MB)	Rpeak (GFLOPS)	Rmax (GFLOPS)	% Peak
IvyBridge	E5-2695 v2	2.4	12	24	115	30	461	458	99.35
IvyBridge	E5-2670 v2	2.5	10	20	115	25	400	389	97.25
Haswell	E5-2697 v3	2.6	14	28	145	35	1160	934	80.52
Haswell	E5-2698 v3	2.3	16	32	135	40	1180	943	79.92
Broadwell	E5-2690 v4	2.6	14	28	135	35	1164	1073	92.18
Broadwell	E5-2697 v4	2.3	18	36	145	45	1324	1186	89.58



Hewlett Packard Enterprise

HPL Performance from a Haswell Cluster





System: **BL460c Gen9, Intel E5-2680 v3, 2.5 GHz, 2P/24C, 128 GB (DDR4-2133 MHz) Memory, RHEL 6.5, IB/FDR 1:1, Intel MPI 5.0.1, Intel Composer XE 15.0.0, Turbo ON, Hyperthreading OFF**

Hewlett Packard Enterprise

Application: High Performance Conjugate Gradient (HPCG)

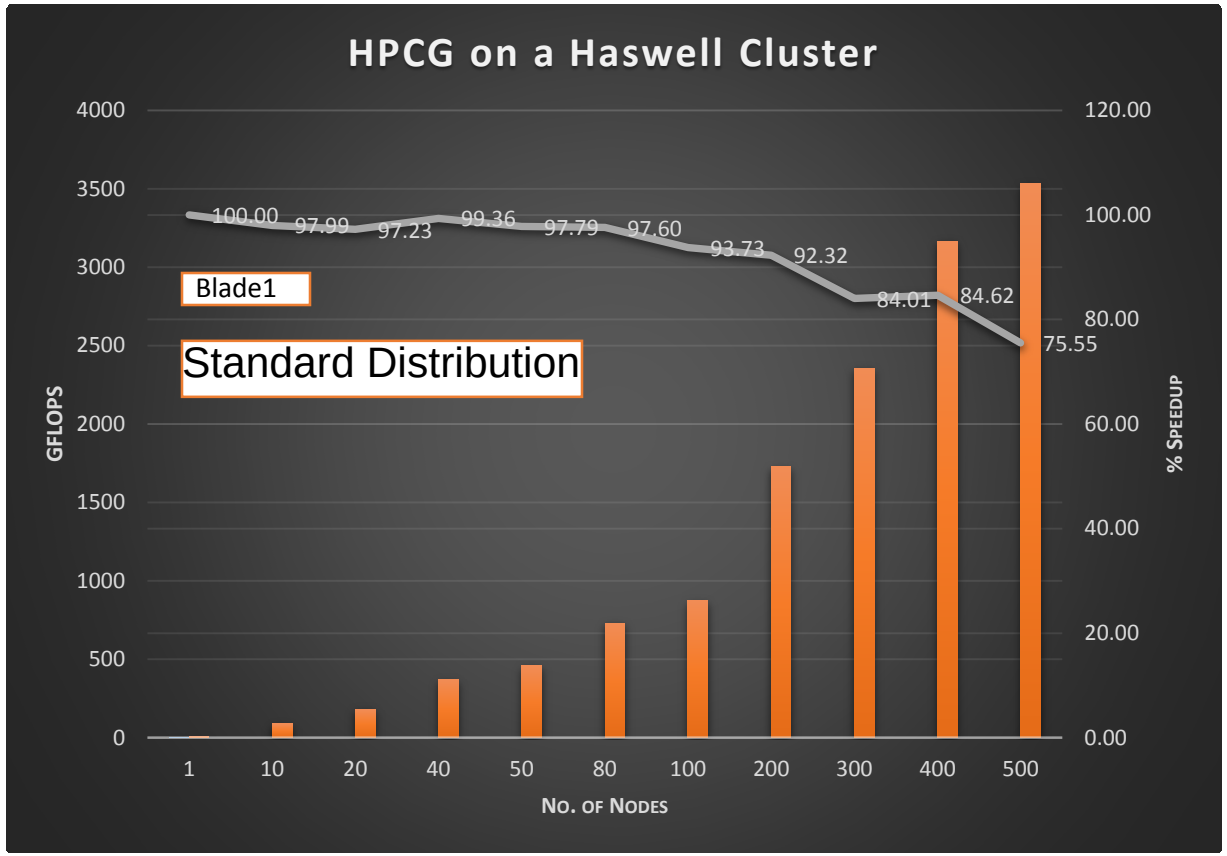
Description:

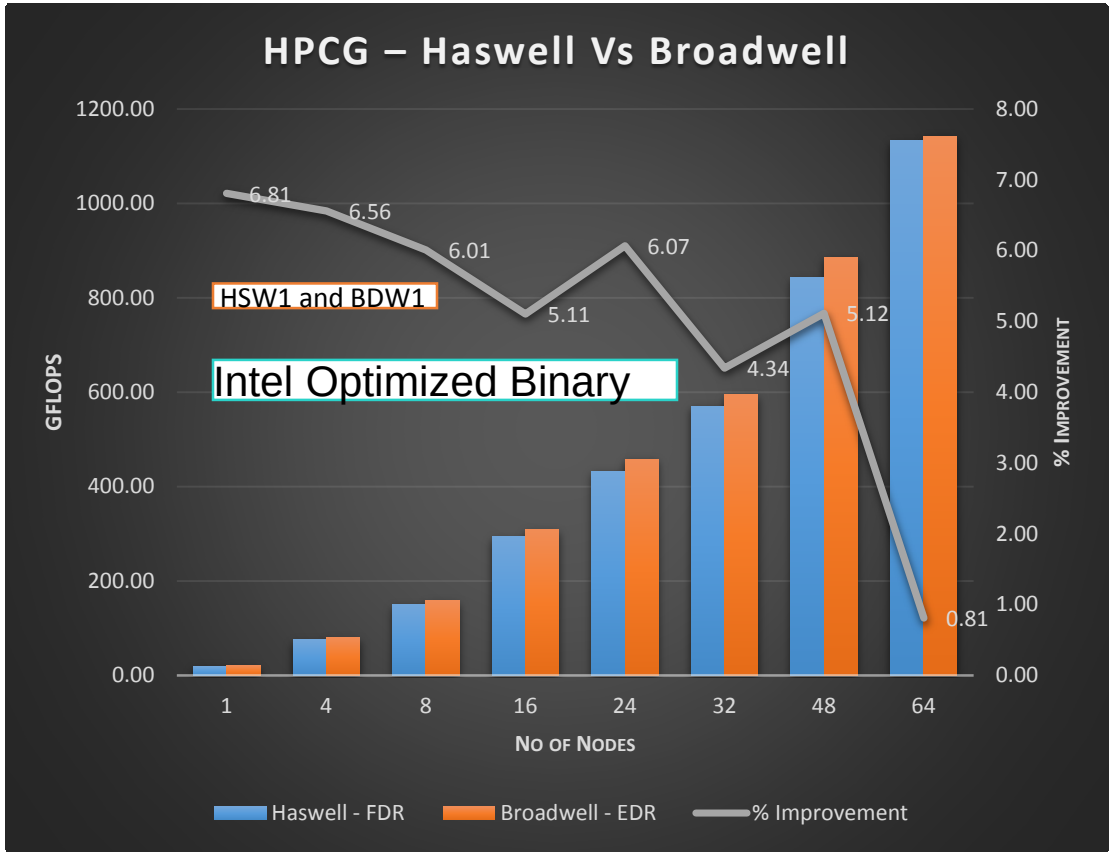
Benchmark designed to create a new metric for ranking HPC systems, complementing the current HPL benchmark. HPCG is designed to exercise computational and data access patterns that more closely match a different and broad set of important HPC applications.

- Supports various operations in a standalone and an unified code.
- Reference implementation is written in C++ with MPI and OpenMP support.
- Driven by multigrid preconditioned conjugate gradient algorithm that exercises the key kernels on a nested set of coarse grids.
- Unlike the HPL, HPCG can be run for predetermined time (input).
- Local domain size (input) for a node is replicated to identify a global domain resulting in near-linear speed-up.
- Performance is measured by GFLOP/s rating reported by the code.
- An Intel optimized HPCG binary was used for this benchmark study.
- Ran the HPCG binary in hybrid mode, MPI processes + OpenMP threads.

Additional details at: http://hpcg-benchmark.org/

High Performance Conjugate Gradient (HPCG)





Blade1: BL460c Gen9, Intel E5-2680 v3, 2.5 GHz, 2P/24C, 128 GB (DDR4-2133 MHz) Memory, RHEL 6.5, IB/FDR 1:1, Intel MPI 5.0.1, Intel Composer XE 15.0.0, Turbo ON, Hyperthreading OFF

HSW1: XL170r Gen9, Intel E5-2698 v3, 2.3 GHz, 2P/32C, 128 GB (DDR4-2133 MHz) Memory, RHEL 6.5, IB/FDR 1:1, Intel MPI 5.0.1, Intel Composer XE 15.0.0, Turbo ON, Hyperthreading OFF

BDW1: XL170r Gen9, Intel E5-2698 v4, 2.2 GHz, 2P/40C, 128 GB (DDR4-2133 MHz) Memory, RHEL 6.6, IB/EDR 1:1, Intel MPI / Intel Compiler 2016.2.181, Turbo ON, Hyperthreading OFF

Application: Graph500

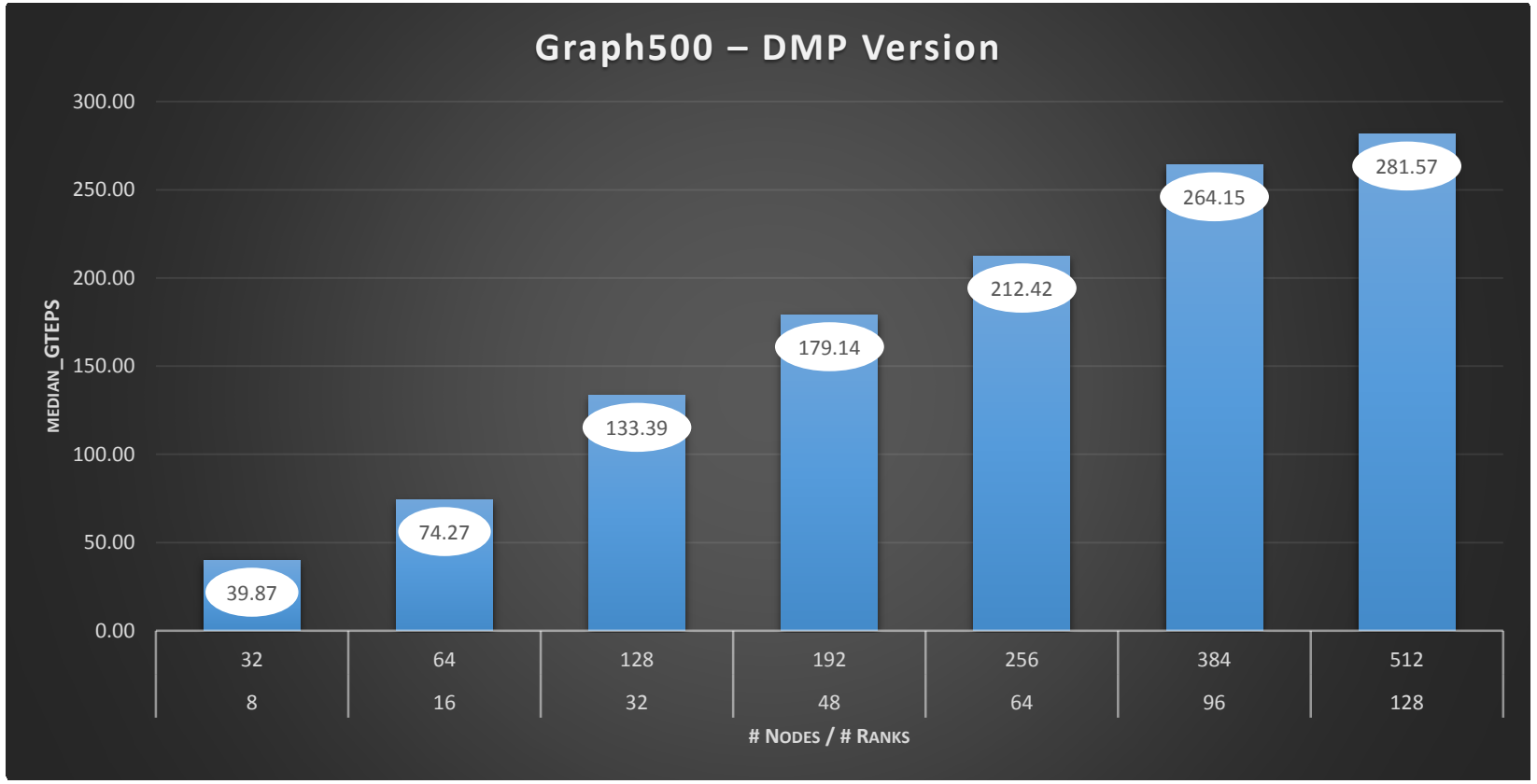
Description:

Benchmark designed to address performance of data intensive HPC applications using Graph Algorithms

- The code generates problem size with a scale (input) creating vertices equal to 10^{scale}
- The performance is measured in TEPS (**T**raversed **E**dges **P**er **S**econd).
- The median_TEPS, in either GTEPS (Giga TEPS) or MTEPS (Mega TEPS) are reported.
- Used a source code optimized for a scale-out (DMP) system by Kyushu University (Japan).
- Application is written in C language.
- Compiled using GNU compiler, gcc.
- Code automatically detects the no. of processors and cores and runs optimally.
- No external placement and binding by the user are needed.
- Needs large memory foot-print to run very large scale problem.

Additional details at: http://www.graph500.org/

Graph500 Performance from a Haswell Cluster



System	Type	Architecture / Chip	CPU	Clock (GHz)	# processors/node	total # cores/node	Memory (GB)	OS	Interconnect	Memory Details	MPI	File System
Hestia	BL460c Gen9	Intel64 / Haswell	E5-2697 v3	2.60	2	28	128	RHEL 6.6	IB / FDR (Connect-IB)	8x16 GB, 2R, DDR4 @ 2133 MHz	Intel MPI 5.0.2.044	NFS

Hewlett Packard Enterprise

Application: Weather Research and Forecasting (WRF)

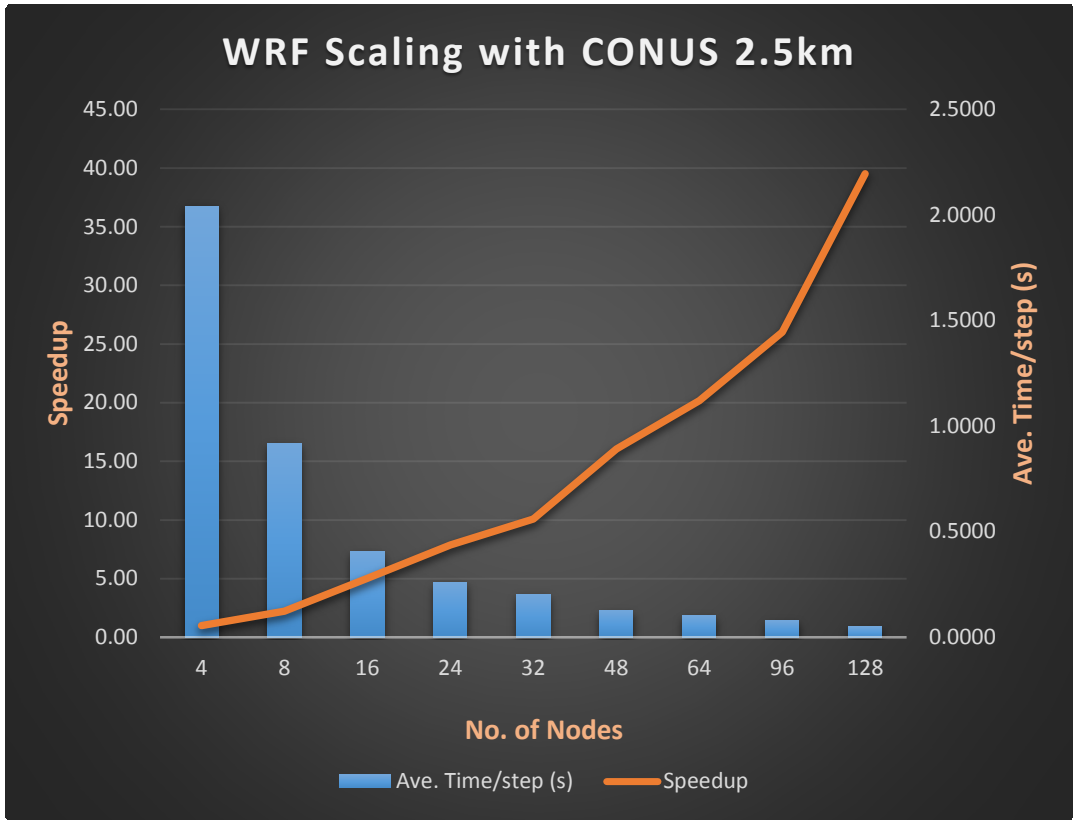
Description:

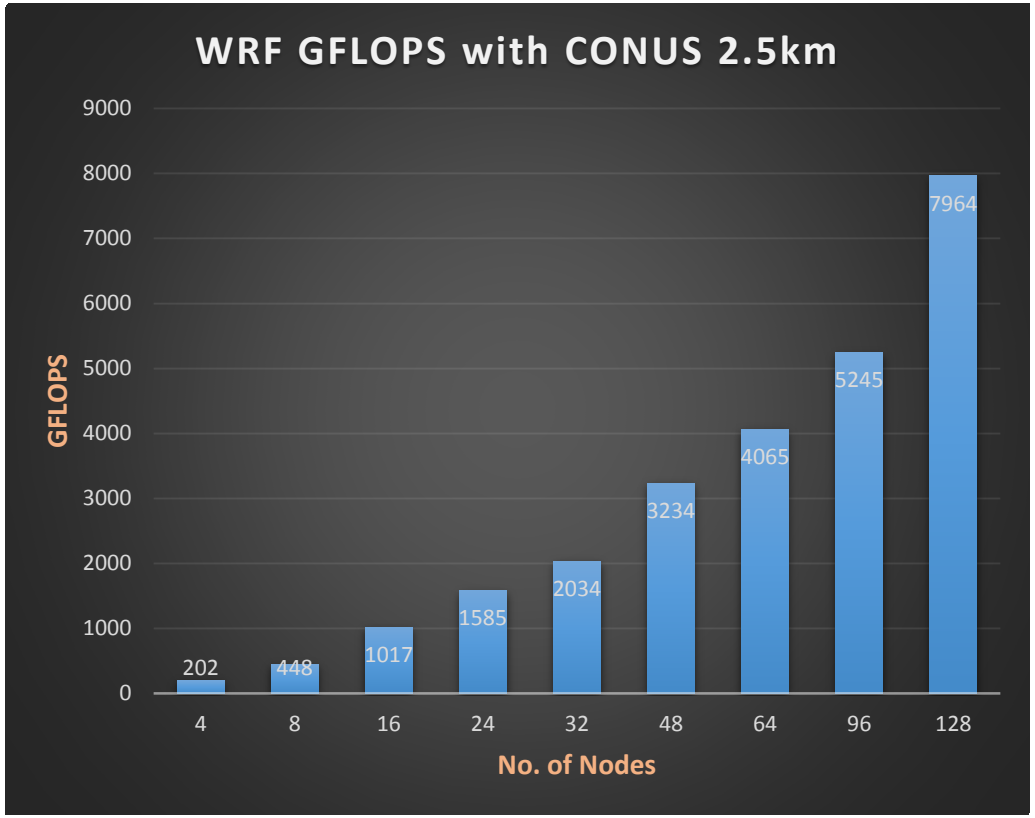
WRF is a Numerical Weather Prediction (NWP) model designed to serve both atmospheric research and operational forecasting needs. NWP refers to the simulation and prediction of the atmosphere with a computer model, and WRF is a set of software to accomplish this.

- The code was jointly developed by NCAR, NCEP, FSL , AFWA, NRL, Univ. of Oklahoma and FAA.
- WRF is freely distributed and supported by NCAR.
- Offers two dynamical solvers: WRF-ARW (Advanced Research WRF) and WRF-NMM (Nonhydrostatic Mesoscale Model).
- Capabilities to mix and match modules to simulate various atmospheric conditions and couple with other NWP models (Ocean Modeling codes)
- Can accommodate simulation with nested data domains, coarse to very fine grids in a single simulation.
- Popular data sets to port and optimize are: CONUS12 and CONUS2.5 (available at NCAR).
- Options to use dedicated processors for I/O (quilting) and various layout of processors (tiling) exist.

Additional details at: http://www.wrf-model.org/index.php

WRF (v 3.8.1) Results with CONUS 2.5km Data Set





System: XL170r Gen9, Intel E5-2699 v4, 2.2 GHz, 2P/44C, 256 GB (DDR4-2133 MHz) Memory, RHEL 6.7, IB/EDR 1:1, Intel Composer XE and Intel MPI (2016.3.210), Turbo ON, Hyperthreading OFF

Hewlett Packard Enterprise

Application: Clover Leaf

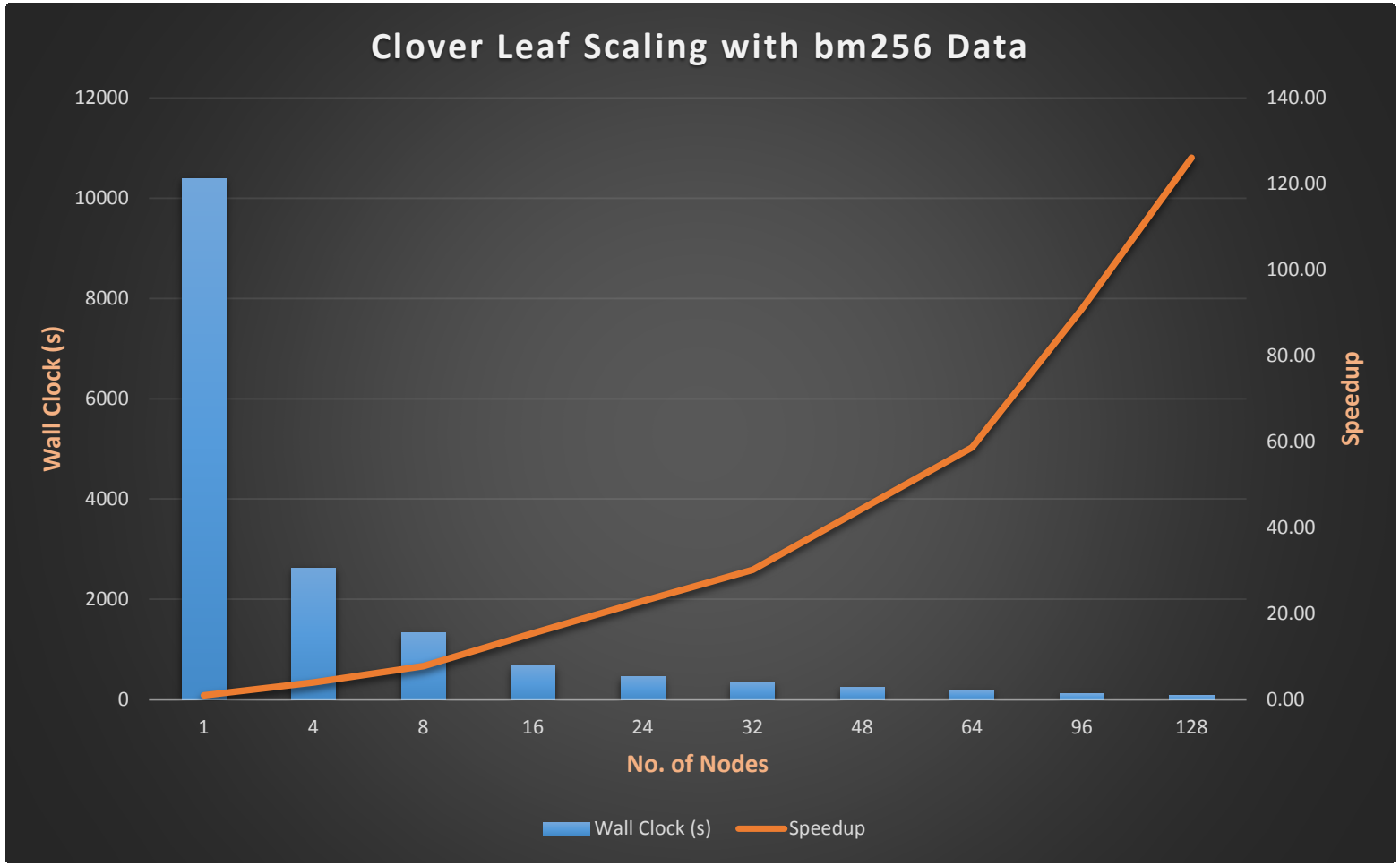
Description:

Clover Leaf is an open-source Computational Fluid Dynamics (CFD) code developed and distributed by UK Mini-Application Consortium (UK-MAC).

- Solves compressible Euler equations on a Cartesian grid using explicit second-order accurate method.
- Uses a ‘kernel’ (low level building block) approach with minimal control logic to increase compiler optimization.
- Supports for accelerators (using both OpenACC and OpenCL) available.
- Scarifies memory (saving intermediate results and than re-computing) to improve performance.
- Available in two flavors, in 2-Dimenional (2D) and 3-Dimensional (3D) modeling.
- Rugged and easy to port and run and good candidate for evaluating and comparing systems.
- Available large no. of data sets for 2D and 3D models with control of run times, few seconds to hours.

Additional details at: http://uk-mac.github.io/CloverLeaf/

Clover Leaf (3D) Results with bm256 Data Set



System: XL230a Gen9, Intel E5-2697A v4, 2.6 GHz, 2P/32C, 128 GB (DDR4-2400 MHz) Memory, RHEL 7.2, IB/EDR 1:1, Intel Composer XE and Intel MPI (2016.3.210), Turbo ON, Hyperthreading OFF

Conclusions

Conclusions

- HP is No. 1 vendor in HPC and Cluster Solutions
- Configure and tune the system first
- Check the system details (processor, clock, memory and BIOS settings)
- Investigate compiler and flags that best suit your application
- Profile the application and optimize further for boosting performance
- Explore and decide on the right interconnect and protocols
- Take advantage of tools and commands to improve performance
- Run the application the right way (environment, placement etc.)
- Choose the right file system (local disk, NFS, Lustre, IBRIX etc.)
- Settle on an environment that is best for your application, time and value
- Never assume and always check the cluster before benchmarking

Hewlett Packard
Enterprise

Thank you

logan.sankaran@hpe.com