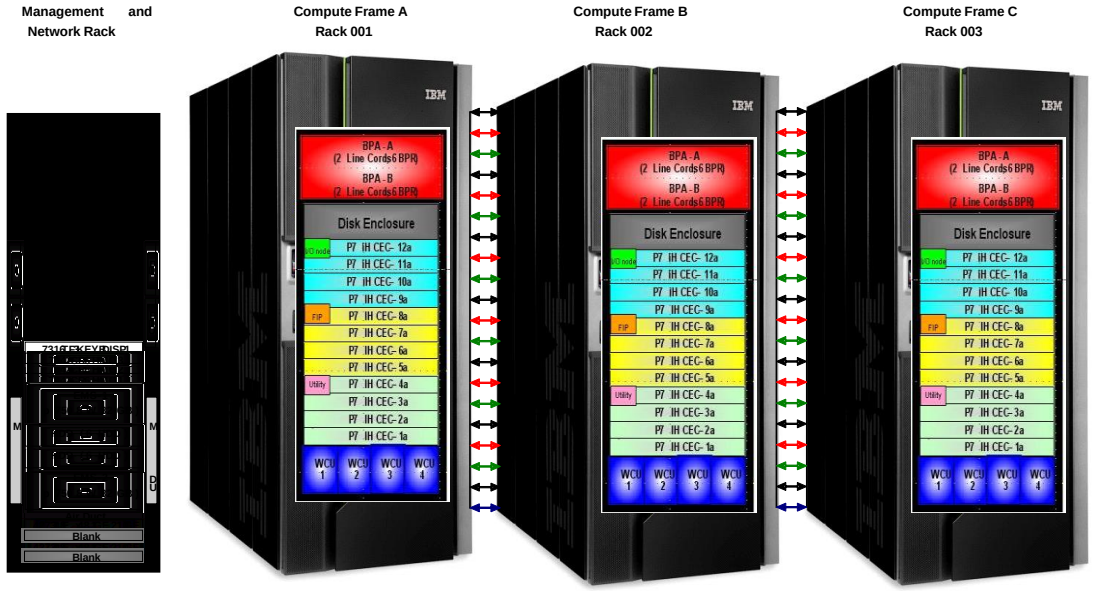
Scaling and performance of a high I/O WRF configuration on three different parallel supercomputers

Zaphiris Christidis (Lenovo), James Abeles (IBM), Min Wei (CMA)

Outline

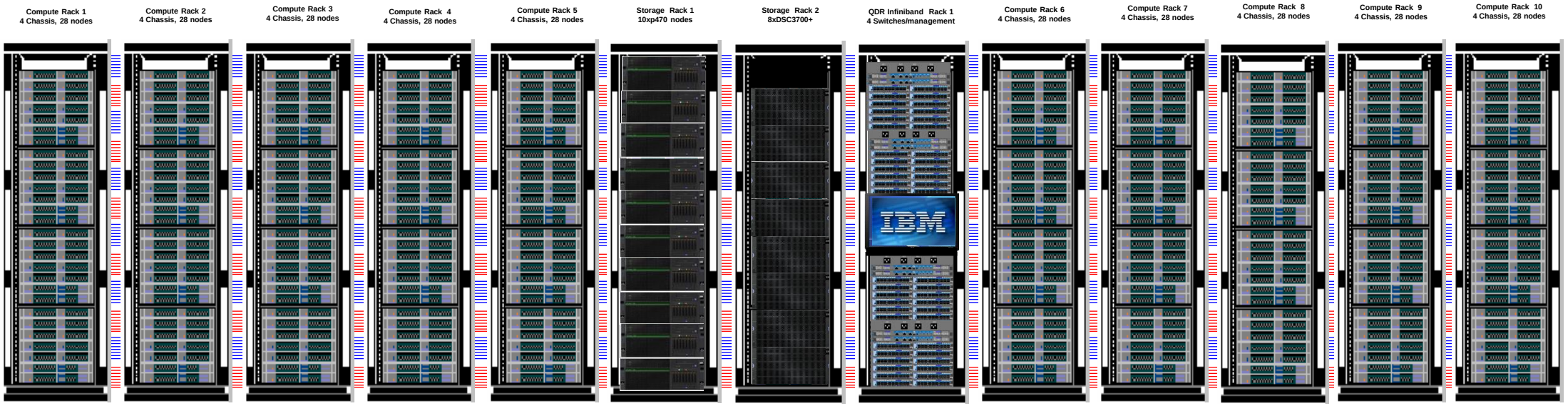
- Parallel System description.
  - *p775, p460 and dx360M4, Hardware and Software*
- Compiler options and libraries used.
- WRF tunable parameters for scaling runs.
  - *nproc_x, nproc_y, numtiles, nio_groups, nio_tasks_per_group*
- WRF I/O
  - *Serial and Parallel netcdf libraries for data I/O.*
- WRF runtime parameters for scaling runs.
  - *SMT, MPI task grids, OpenMP threads, quilting tasks.*
- WRF performance components for scaling runs.
  - *Computation, Communication, I/O, load Imbalance.*
- WRF scaling run results on p775, p460 and dx360M4.
- Conclusions.

IBM POWER 7 p775 system (P7-IH)



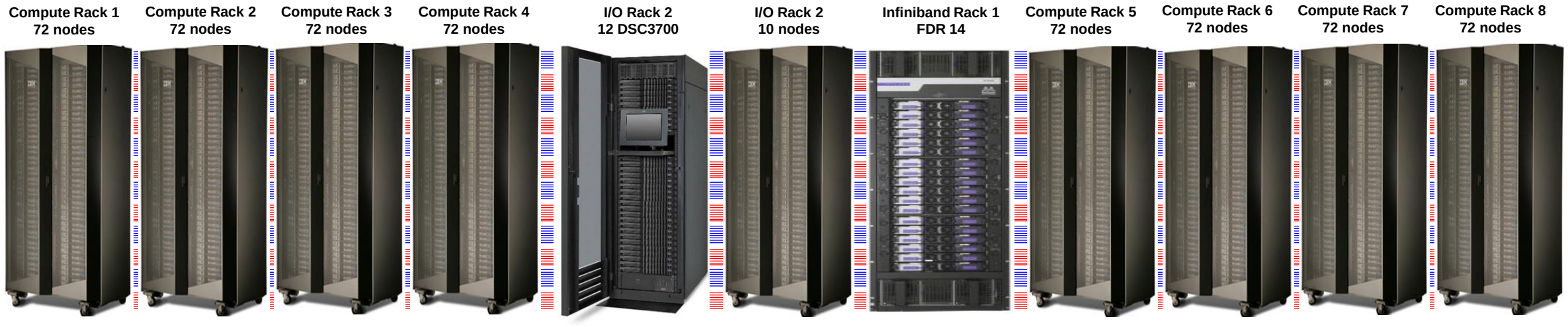
- Similar to the old ECMWF system, Water Cooled
  - *Rack: 3 Supernodes; Supernode: 4 drawers; Drawer: 32 POWER7 chips; Chip: 8 POWER7 cores.*
  - *32 Cores/node @ 3.86 GHz, 256 GB/node, 8 GB/core, 1024 cores/drawer, 3072 cores/rack,*
  - *L1: 32 KB instruction/32 KB data; L2: 256 KB/core; L3: 4 MB/core, Dual memory channel*
  - *Diskless node, Torrent POWER7 chip, GiGE adaptors on service nodes*
  - *HFI fibre network, 4 x storage (GPFS) nodes, 3 x Disk Enclosures.*
  - *AIX 7.1, XLF14, VAC 12, GPFS, IBM Parallel Environment, LoadLeveler*
  - *MPI profile Library, Floating Point Monitor, Vector and Scalar Mass Libraries*

IBM POWER 7 p460 Pureflex system (Firebird)



- CMA system partition, Air Cooled, Rear Door Heat Exchangers.
  - *Rack: 4 chassis; Chassis: 7 nodes; Node: 4 POWER7 chips; Chip: 8 POWER7 cores*
  - *32 Cores/node @ 3.55 GHz,128 GB/node, 4 GB/core, 224 cores/chassis, 896 cores/rack,*
  - *L1: 32 KB instruction/32 KB data; L2: 256 KB/core; L3: 4 MB/core, single memory channel*
  - *2 HD per node, 2 QDR Dual port Infiniband adaptors, GiGE*
  - *Dual Rail Fat tree Infiniband network, 10 x p740 storage (GPFS) nodes, 8 x DSC3700 devices.*
  - *AIX 7.1, XLF14, VAC 12, GPFS, IBM Parallel Environment, LoadLeveler*
  - *MPI profile Library, Vector and Scalar Mass Libraries*

IBM iDataplex dx360 M4 system (Sandybridge)



- NCEP WCOSS system partition, Air Cooled, Rear Door Heat Exchangers
  - *Rack: 72 Nodes; Node: 2 Intel dx360 m4 sockets; Socket: 8 Intel E5-2670 Sandybridge cores.*
  - *16 Cores/node @ 2.6 GHz, 32 GB/node, 2 GB/core, 1152 cores/rack,*
  - *L1: 32 KB instruction/32 KB data; L2: 256 KB/core; L3: 20 MB shared per 8 cores*
  - *1 HD per node, Mellanox connect X - FDR, 10GiGE ports*
  - *Mellanox IB4X FDR full bisection Fat Tree network, 10 x 3650 M4 storage nodes, 20 x DSC3700*
  - *RHEL6.2, Intel FORTRAN 13, C 11, GPFS, IBM Parallel Environment, Platform LSF*
  - *MPI profile Library*

Tested Systems Summary

	Core frequncy (GHz)	Memory (GB/core)	Cores tested	Vector	SMT/HT settings	IC Fabric	OS	Storage	Compilers	Parallel Envrnmnt	Queueing systems	Libraries
p775	**3.86**	**8** (All dimms occupied)	**6144**	**VSX** only for intrinsics	**SMT4** (SMT2 to 2048 cores)	**HFI**	**AIX 7.1**	**GPFS** **4 NSD** 3 Disk Enclsr	**IBM XL** Compilers (AIX v14)	**IBM PE** for AIX	**Load Leveler**	**MPI** Profile, MASS, Hardware Perfrmnce Monitor
p460	**3.55**	**4** (All dimms occupied)	**8192**	**VSX** Only for Intriscics	**SMT4** (SMT2 to 512 cores)	**QDR** Dual-Rail Infiniband Fat Tree	**AIX 7.1**	**GPFS** **10 NSD** 8 DSC3700	**IBM XL** Compilers (V14)	**IBM PE** for AIX	**Load Leveler**	**MPI** Profile, MASS
dx360M4	**2.6**	**2** (All dimms occupied)	**6144**	**AVX** Every where	**HT** (Not used)	**FDR** Single Rail Infiniband Fat Tree	**RHEL 6.2**	**GPFS** **10 NSD** 20 DSC3700	**Intel** Studio 13 Compilers	**IBM PE** for linux	**LSF**	**MPI** Profile

WRF v3.3 Compiler Options

Identical WRF code was used	COMPILER Options	Libraries
p775	FCOPTIM = -O3 -qhot -qarch=pwr7 -qtune=pwr7 FCBASEOPTS = -qsmp=omp -qcache=auto -qfloat=rsqrt	netcdf, pnetcdf, massp7_simd, massvp7, mpihpm
p460	FCOPTIM = -O3 -qhot -qarch=pwr7 -qtune=pwr7 FCBASEOPTS = -qsmp=omp -qcache=auto -qfloat=rsqrt	netcdf, pnetcdf, massp7_simd, massvp7, mpitrace
dx360M4	FCOPTIM = -O3 -xAVX -fp-model fast=2 -ip FCBASEOPTS = -ip -fno-alias -w -ftz -no-prec-div -no-prec-sqrt -align all -openmp	netcdf, pnetcdf, mpitrace

WRF tunables for scaling runs

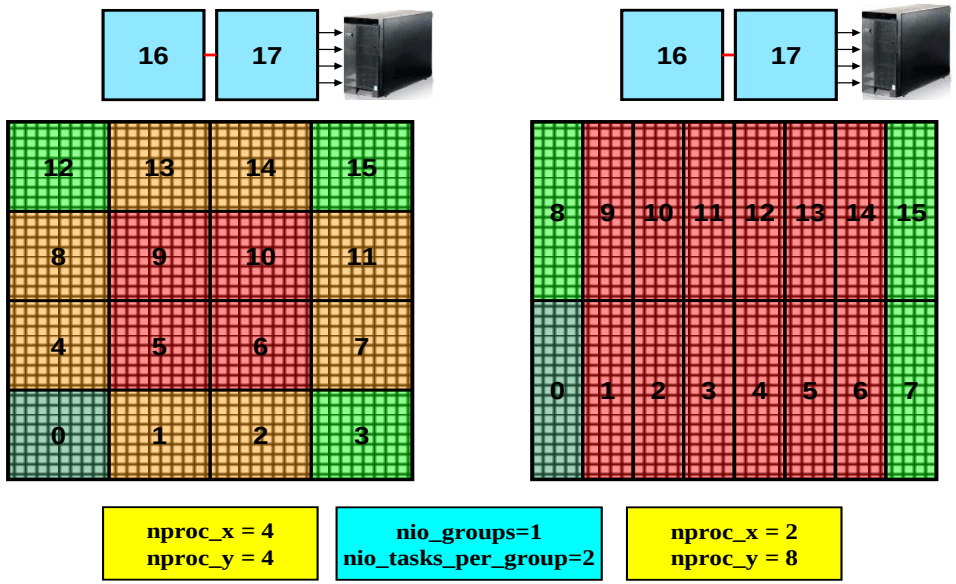
- WRF run specifics as defined in namelist.input:
  - *5km horizontal resolution, 6 sec time step, 12-hour forecast.*
  - *2200 X 1200 X 28 grid points.*
  - *One output file per forecasting hour.*
  - *Four Boundary reads every three forecast hours.*
- Same WRF tunables were used for every system, based on selections that yielded optimal performance on the p775 system.
  - ***nproc_x:*** *Logical MPI task partition in x-direction.*
  - ***nproc_y****: Logical MPI task partition in y-direction.*
  - ***numtiles:*** *Number of tiles that can be used in OpenMP.*
  - *nproc_x X nproc_y = number of MPI tasks. Critical in comparing communication characteristics.*
- SMT was used only on the IBM POWER systems.
  - *SMT was employed on POWER systems by using two OpenMP threads per MPI task.*
  - *SMT2 was used for runs with less than 2048 cores on p775 (Best performance).*
  - *SMT2 was used for runs with less than 512 cores on p460 (Best performance).*
  - *Hyper-threading was not beneficial on dx360M4 system (not used).*

WRF I/O

- I/O reading and writing of WRF variables.
  - *13 I/O write steps, each writing a 7.5 GB file.*
  - *1 Read for the Initial conditions (6.74 GB file).*
  - *4 Reads for the boundary conditions (1.47 GBs each).*
  - *Data ingests cannot be done asynchronously.*
- Parallel netcdf was used for data ingests (MPI-IO) for all scaling runs
  - *Read option 11 for initial and boundary data.*
- I/O netcdf quilting was used to write data files.
  - *Assign the same I/O tasks and groups on all three systems for each of the scaling runs.*
  - *Last I/O step is done synchronously, since WRF computations terminate.*
  - *Quilting I/O times on WRF timers report I/O synchronization time only.*
  - *I/O is done by quilting tasks on the I/O subsystem while compute tasks compute.*
- I/O Parallel netcdf quilting was not used.
  - *Can further improve I/O writing steps, especially the last I/O step.*
  - *Early WRF version had problems with IBM Parallel Environment and parallel netcdf.*

WRF uniform variables

- Unchanged variables on all systems for the same number of physical cores
  - ***nproc_x; nproc_y, nio_groups, nio_tasks_per_group, numtiles***
- Performance on POWER systems was found to be always better when:
  - ***nproc_x < nproc_y*** *(thin rectrangular MPI task decomposition)*
    - ✓ POWER processors have large non-shared L3 caches, so probably thin decompositions offer better cache utilization.
    - ✓ Out-of-stride data copying from application to system buffers for MPI communication is minimized on thin decompositions.
    - ✓ **O3 –qhot** compiler option introduces vector MASS library calls, which deliver better throughput for long vectors.
- Performance on dx360M4 was better when:
  - ***numtiles > 1*** *even for runs with a single OpenMP thread.*
    - ✓ **numtiles > 1** acts as a cache block mechanism (like NPROMA)
- p775 and p460 are favored against dx360M4
  - *For the testing scenarios if* ***nproc_x < nproc_y.***
  - ***numtiles > 2*** *did not have an effect on performance.*
- Choice of nproc_x, nproc_y, numtiles:
  - *Was based on best performance on the p775*
  - ***numtiles = 4*** *was set as an advantage on dx360M4*



WRF runtime parameters

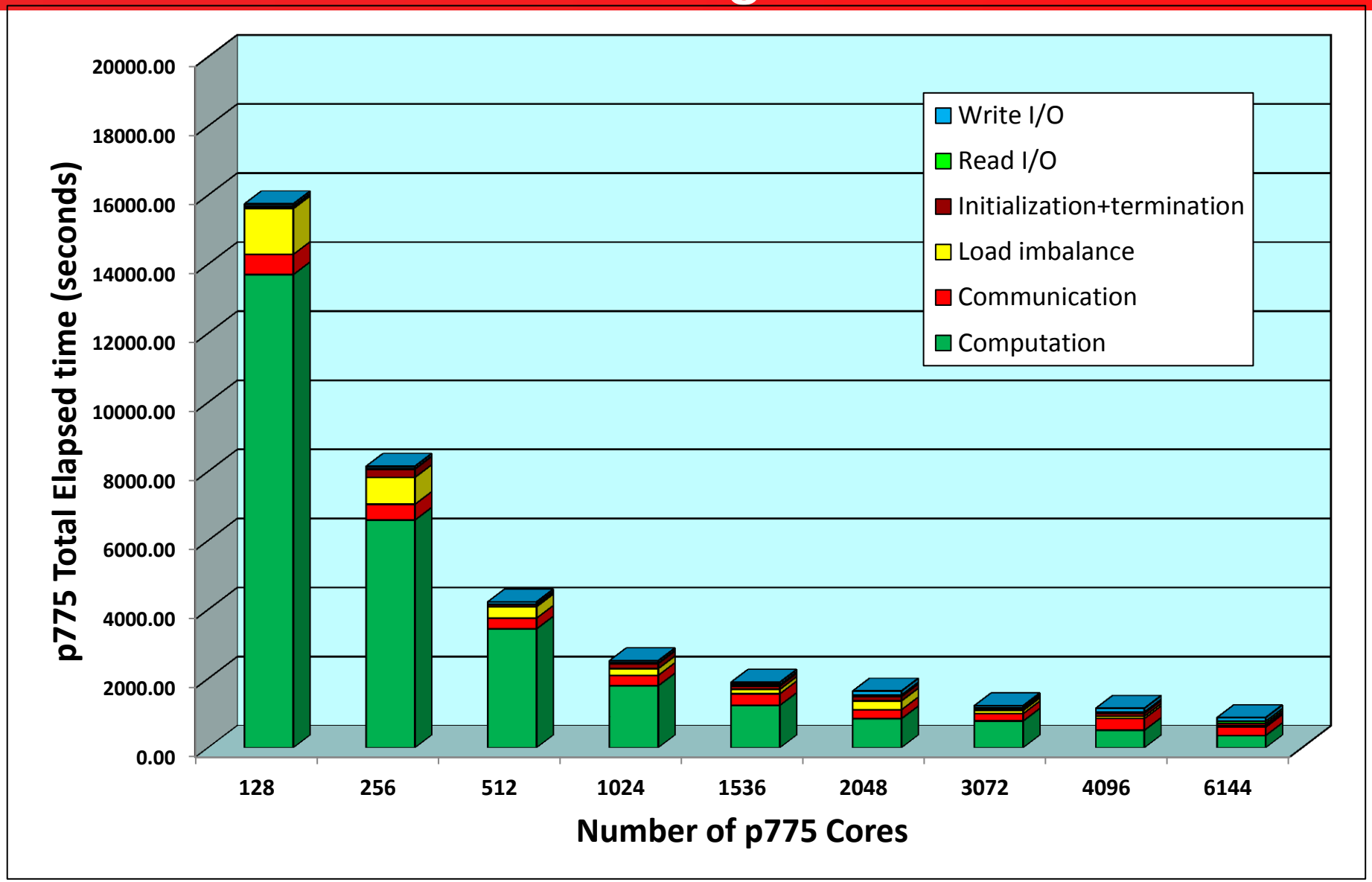
- Task affinity (binding) was used in all test runs.
  - ***SMT/HT*: *ON* – *green background (One extra OpenMP thread), *OFF* – *yellow background.***
  - *Variables: OpenMP threads, numtiles, MPI tasks, nproc_x, nproc_y, nio_groups, nio_tasks_per_group*
  - *Physical/Logical cores = OpenMP_threads * (nproc_x * nproc_y +nio_groups*nio_tasks_per_group)*

Number of p775 nodes	OpenMP Threads	Number of p460 nodes	OpenMP Threads	Number of dx360 nodes	OpenMP Threads	numtiles	Number of cores	MPI Tasks	nproc_x x nproc_y	nio_groups x nio_tasks_per_group
4	2	4	2	8	1	4	128	124	4x31	1x4
8	2	8	2	16	1	4	256	252	6x42	1x4
16	2	16	2	32	1	4	512	504	8x63	1x8
32	4	32	2	64	2	4	1024	506	11x46	1x6
48	4	48	2	96	2	4	1536	760	19x40	1x8
64	4	64	2	128	2	4	2048	1020	20x51	1x4
96	4	96	4	192	4	4	3072	760	19x40	1x8
128	4	128	4	256	4	4	4096	1020	17x60	1x4
192	4	192	4	384	4	4	6144	1530	18x85	1x6

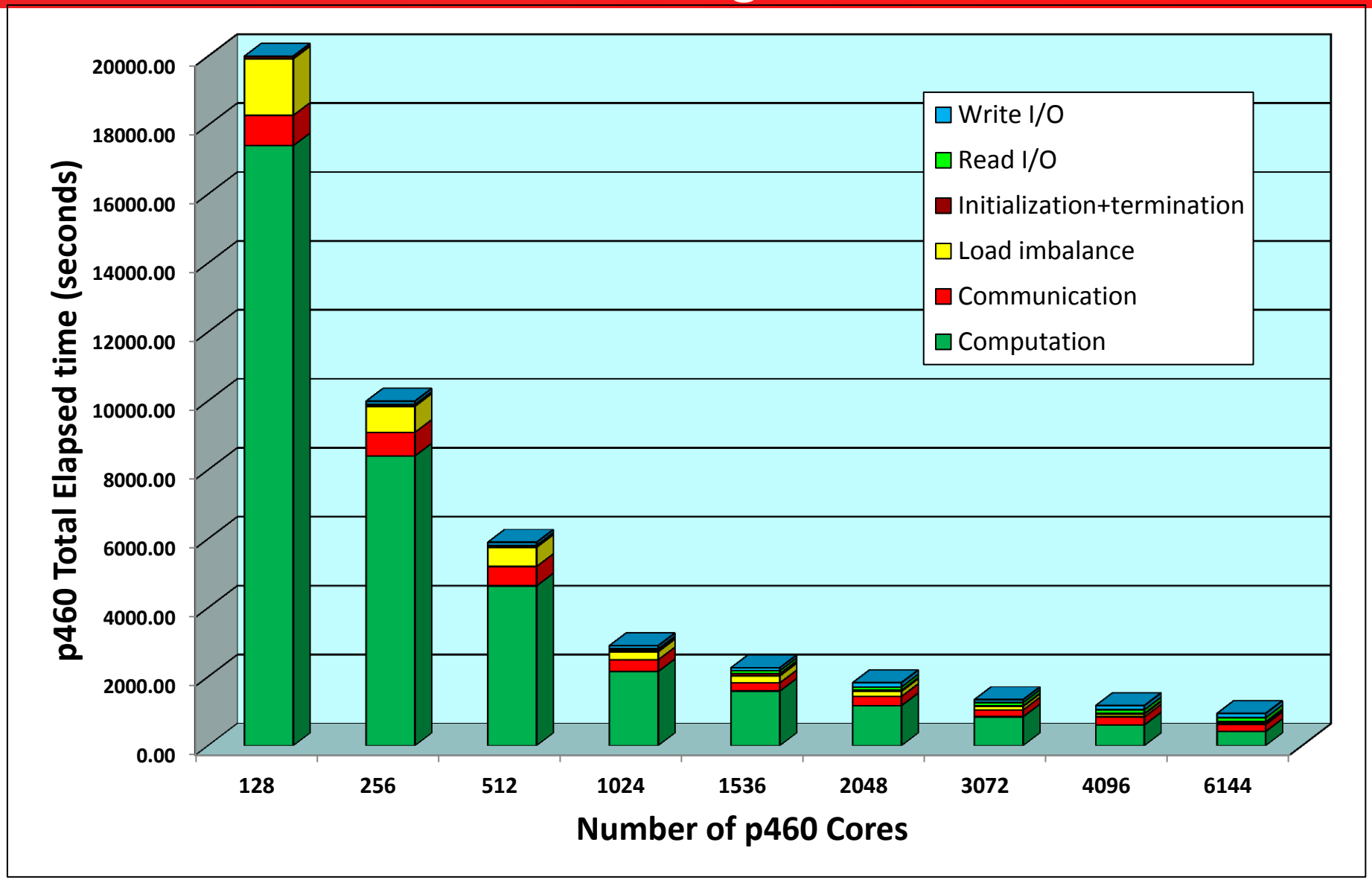
WRF Run statistics

- Runs on p775 were done with the MPIHPM library:
  - *Collect Hardware performance monitor data (small overhead).*
    - ✓ Scale HPM data to p460 and dx360M4 systems by frequency ratios)
    - ✓ Estimate Sustained GFLOP rates on all systems.
    - ✓ System Peak rate = (number of cores x 8 x core frequency).
  - *Collect MPI communication statistics.*
- Runs on p460 and dx360M4 were done with the MPITRACE library:
  - *Collect MPI communication statistics.*
- MPI communication from trace libraries can help estimate:
  - *Communication: (**minimum communication among all MPI tasks involved**).*
  - *Load Imbalance: (**median communication – minimum communication**).*
- Accumulation of internal WRF timers can help estimate:
  - *Read I/O times (**initial file read time + boundary read times**).*
  - *Write I/O times (**I/O Write quilting time from synchronization + Last I/O Write time step**).*
  - *Last I/O write step: ~(**total elapsed time – total time from internal timers**).*
  - *Total Computation (**Pure computation + communication + Load imbalance**).*

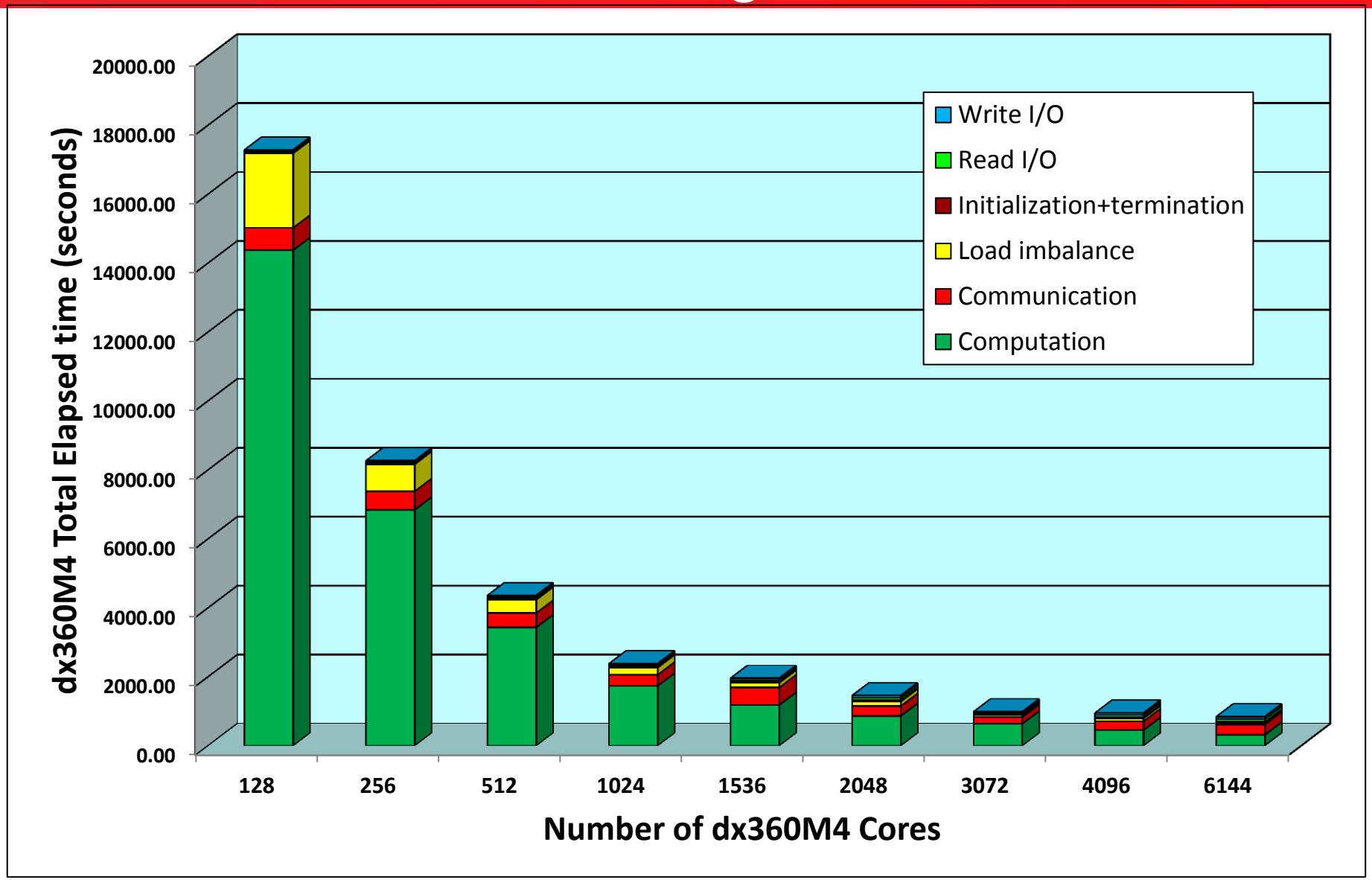
WRF Scaling Results



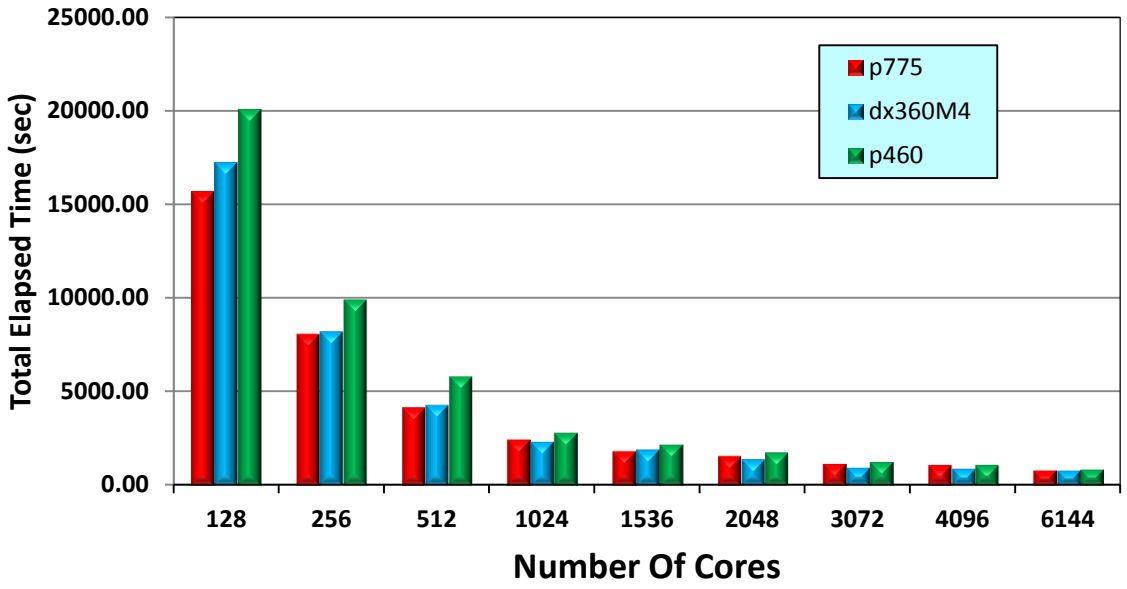
WRF Scaling Results

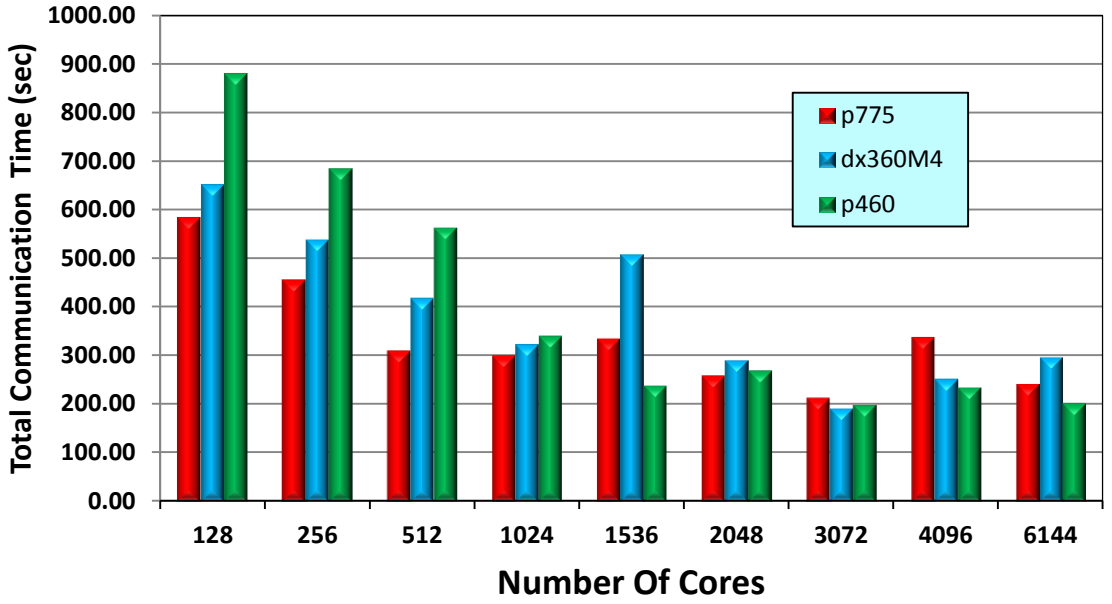


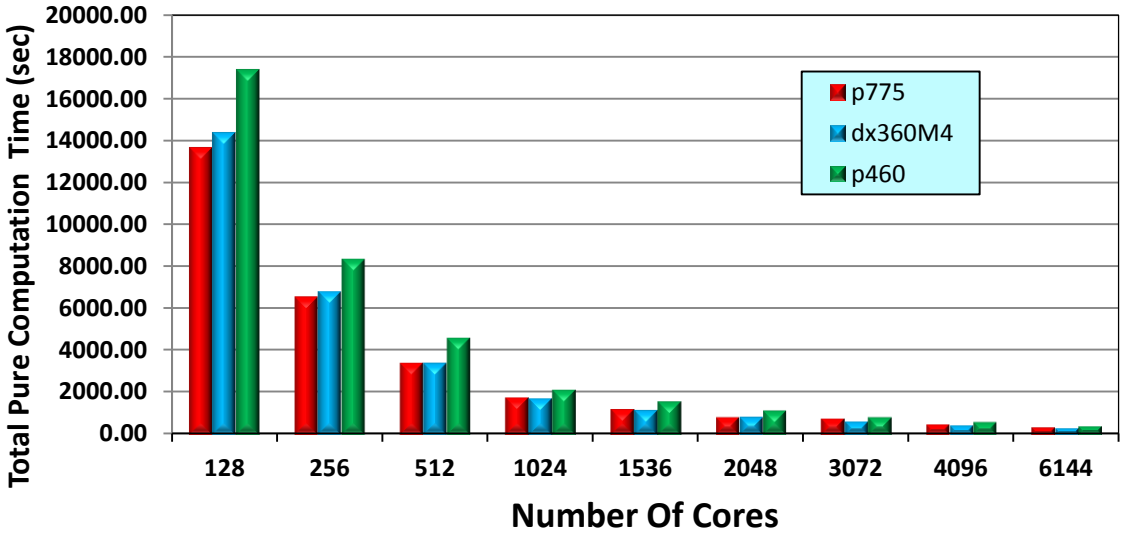
WRF Scaling Results

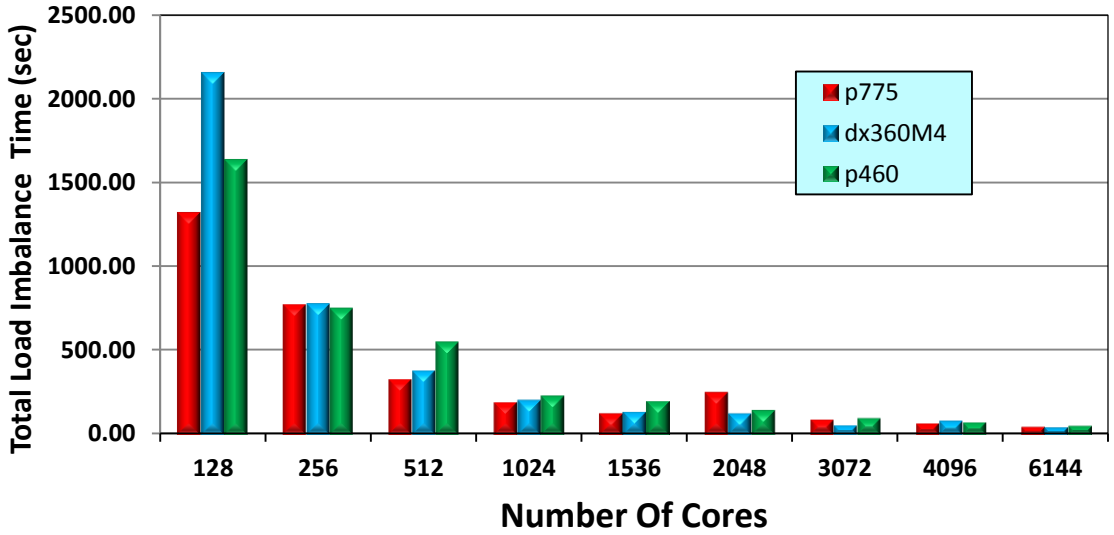


WRF Run statistics

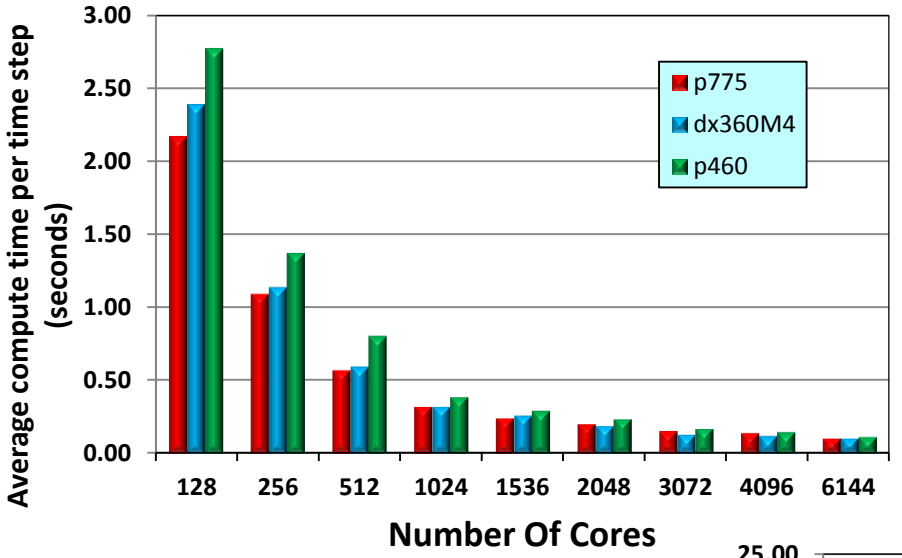


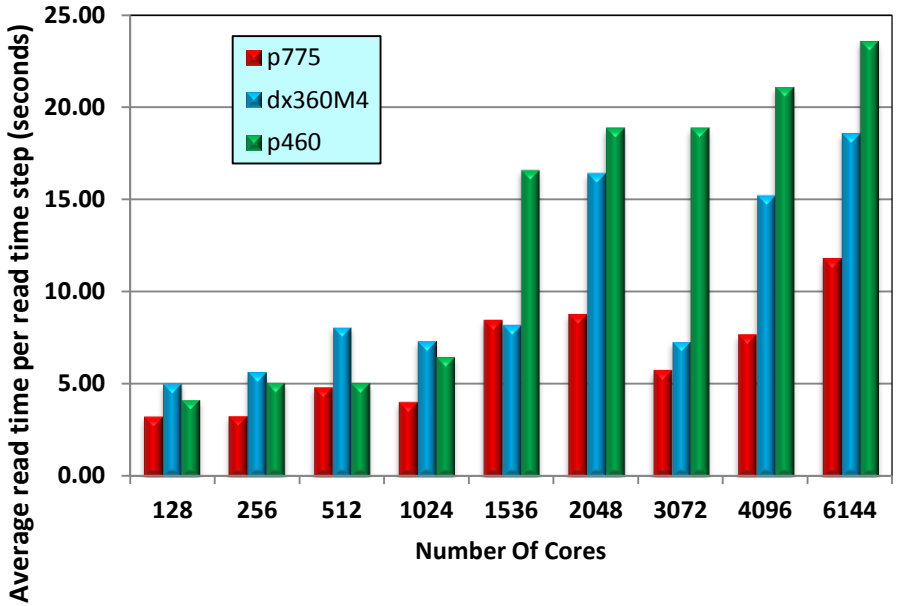


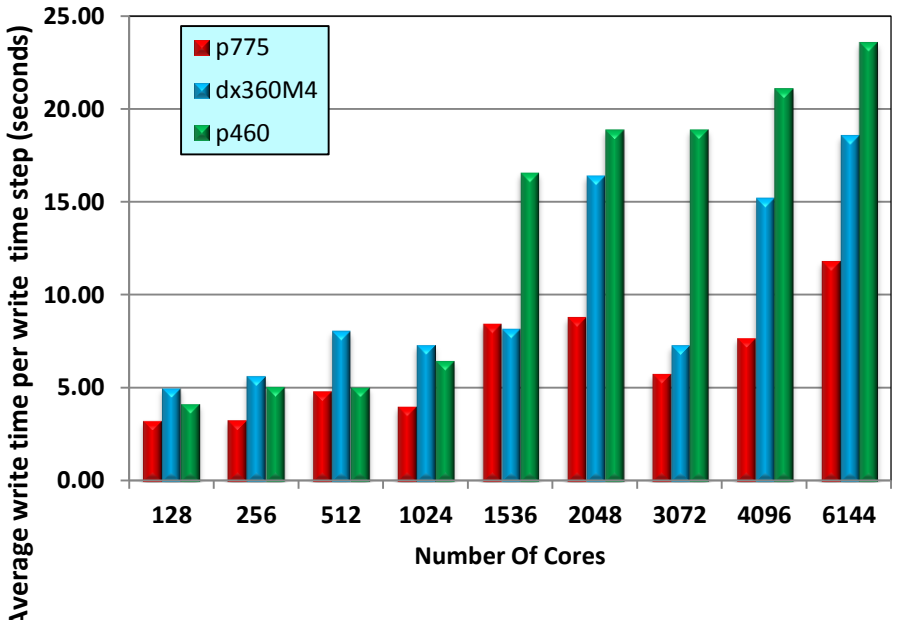




WRF Run statistics

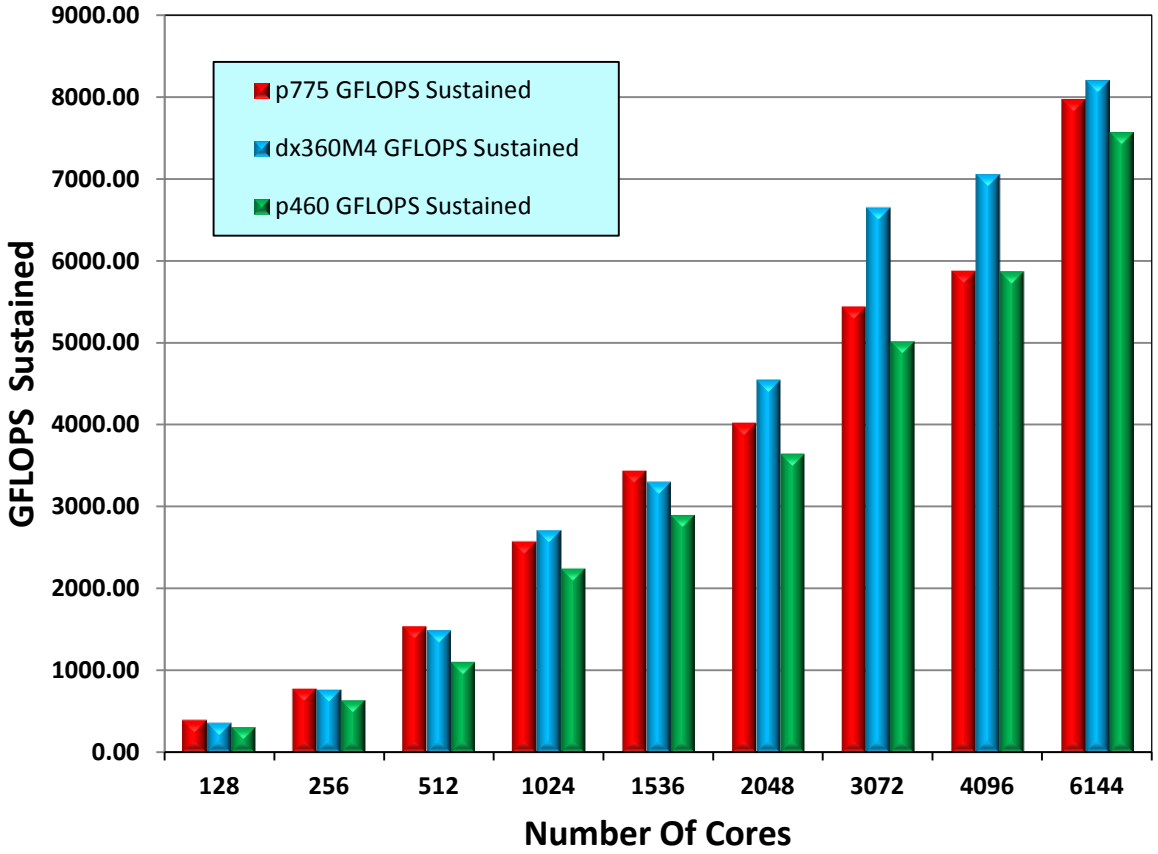


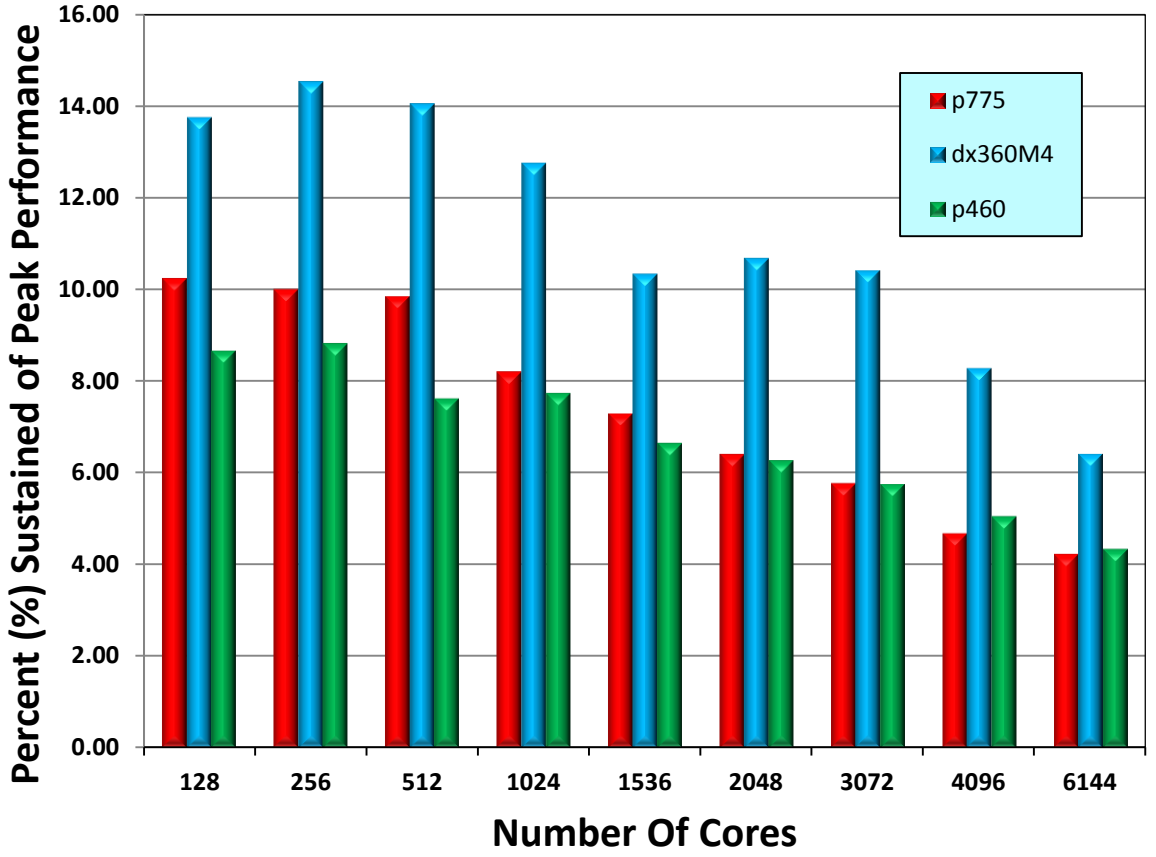




WRF GFLOP Rates

Peak GFLOPS = Number of cores X 8 x Core frequency.
p775 Sustained GFLOPS = (10^{-9} / p775_run_time) * (PM_VSU_1FLOP+2*PM_VSU_2FLOP+4*PM_VSU_4FLOP+8*PM_VSU_8FLOP)
p460 Sustained GGLOPS = p775 Sustained GFLOPS * (p775_run_time / p460_run_time)
dx360M4 Sustained GFLOPS = p775 Sustained GFLOPS * (p775_run_time / dx360M4_run_time)





Conclusions

- WRF scales and performs well on all tested systems.
- Quilting I/O with netcdf works very well on all systems.
- Parallel netcdf for data ingest improves data read times.
- WRF is a popular single precision Code.
  - *It runs very well on dx360M4 system.*
    - ✓ **-xAVX** works very well.
    - ✓ Intel Compilers do a great job producing optimal and fast binaries.
    - ✓ **numtiles** ~cache block parameter for additional performance.
    - ✓ Hyperthreading gives no benefit towards overall performance.
    - ✓ Near neighbor communication is handled effectively by FDR IB.
  - *It runs ok on p775 and p460 systems.*
    - ✓ **VSX** does not work well. Code crashes if compiled with **-qsimd**
    - ✓ IBM XL compilers do OK with **–O3 –qhot** (Vector **MASS** library).
    - ✓ Thin rectangular decompositions work ok (caching and vector MASS).
    - ✓ SMT works well on p775, due to available memory-to-core BW.
    - ✓ Near neighbor communication an overkill for p775, but OK for p460.
  - *Performance odds were stacked against dx360M4.*
    - ✓ Runs with 6144 cores and different **nproc_x, nproc_y** yield even better performance.