Improving I/O Performance of the Weather Research and Forecast (WRF) Model

Tricia Balle & Pete Johnsen

CUG, May 2016

Agenda

- **Numeric prediction models become increasingly complex as size and availability of HPC resources increase**
  - Forecast accuracy improves
  - Simulations at grid resolutions < 2km over entire globe can generate terabytes of weather information written frequently over forecast cycle
  - Increased demands on I/O subsystems creates performance bottlenecks
- **WRF offers a range of different I/O options**
  - How do we use the asynchronous and parallel I/O features of WRF to best take advantage of a Cray Lustre parallel file system and Cray MPI-IO?
- **Initial results using DataWarp**
- **Summary and Q&A**

Copyright 2016 Cray Inc.

WRF Background (wrf-model.org)

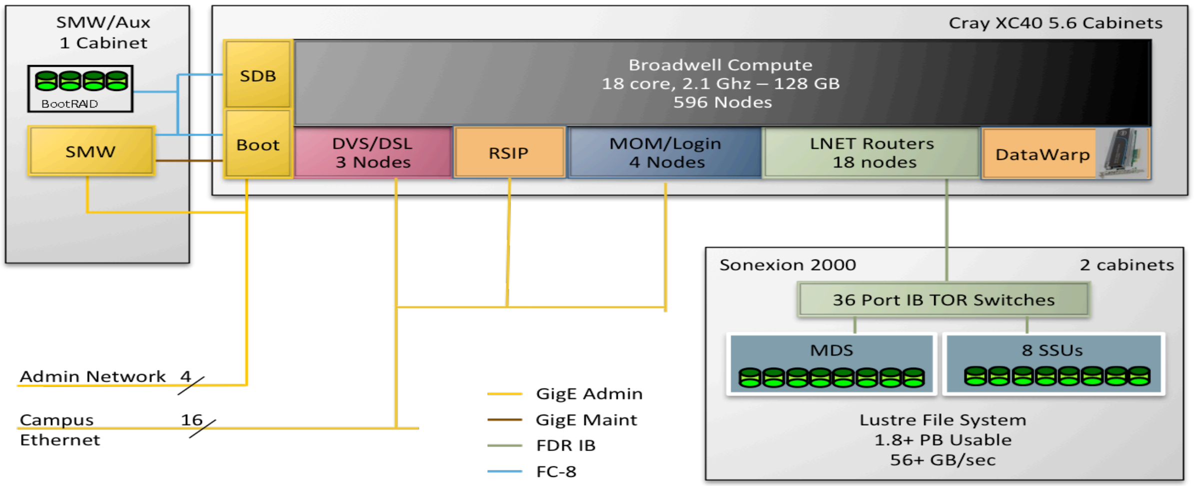
- **Collaborative project by NCAR, NOAA, NCEP, Air Force Weather Agency, NRL, University of Oklahoma, and the FAA**
- **Regional to global scale numerical weather prediction model for both research and operational forecast systems**
- **Suitable for broad range of meteorological applications across scales from meters to thousands of kms**
- **Used by weather agencies all over the world, e.g., by NOAA for primary regional forecast model for 5 days ahead**
- **Open source, over 10,000 registered users**
- **Designed to perform well on massively parallel computers**
  - Uses MPI and OpenMP
  - Written in Fortran90
  - Components have been ported to GPUs and to Intel MIC

Copyright 2016 Cray Inc.

Benchmark System (Cray XC40)

- Intel Broadwell processors (18-core, 2.1ghz)
- 128GB DDR4 (2400mhz) memory per node
- Sonexion 2000 storage, 16 Lustre OSTs
- Also: Cray DataWarp storage (3 nodes)

See paper for more detail

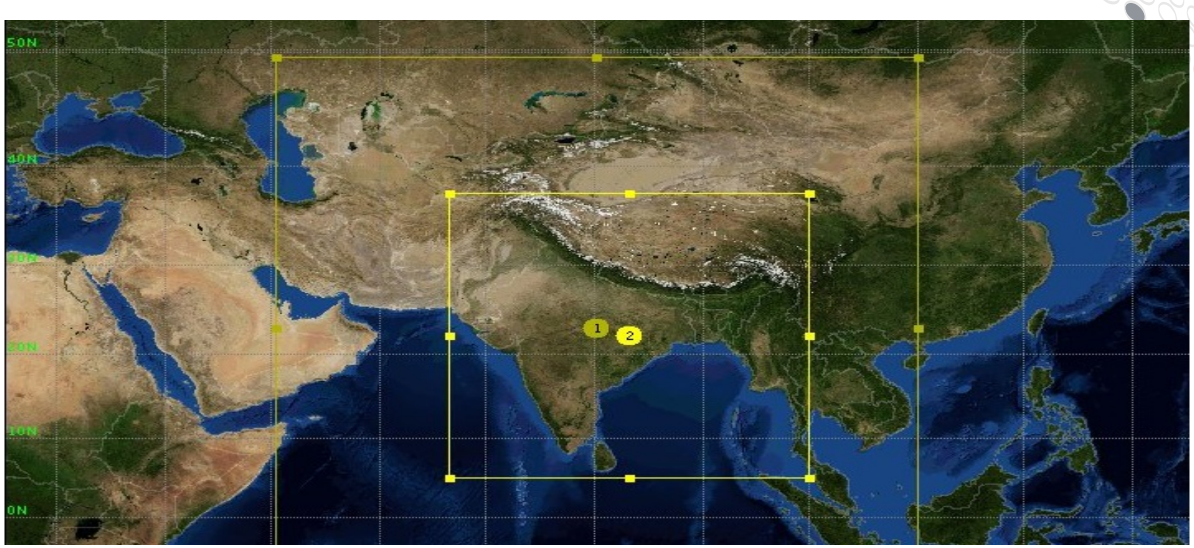


Copyright 2016 Cray Inc.

WRF Benchmark Configuration

Typical high-resolution configuration used by weather services

- Southeast Asia
- Two nested domains, 3km and 1km
  - Dom1: 1770x1986
  - Dom2: 2974x3118
- 28 vertical levels



- 30 minute simulation
- 5 second timestep
- History files written every 15 mins by each domain
  - Dom1: 19.8GB per output step
  - Dom2: 7.5GB per output step

Copyright 2016 Cray Inc.

WRF I/O Implementations....

1. Serial NetCDF (default)

- Set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data
- Common data format used in environmental sciences
- Provided as part of Programming Environment on Cray XC

2. Parallel NetCDF

- Extension of NetCDF that supports parallel I/O
- Collaborative effort from Argonne and Northwestern University
- Implemented using Cray MPI-IO layer

Copyright 2016 Cray Inc.

WRF I/O Implementations (cont.)

3. **Quilt Servers (output only)**
   - I/O servers increasingly common feature in weather/climate codes
   - Asynchronous I/O
   - Assign number of ranks for I/O only in groups
   - Serial NetCDF writes within server groups

4. **Quilt Servers with Parallel NetCDF (output only)**
   - As above but with parallel writes within groups
   - Implemented by Andrew Porter, STFC Daresbury Lab (cf. 2010 CUG paper)

Main focus on output I/O only in what follows....

Copyright 2016 Cray Inc.

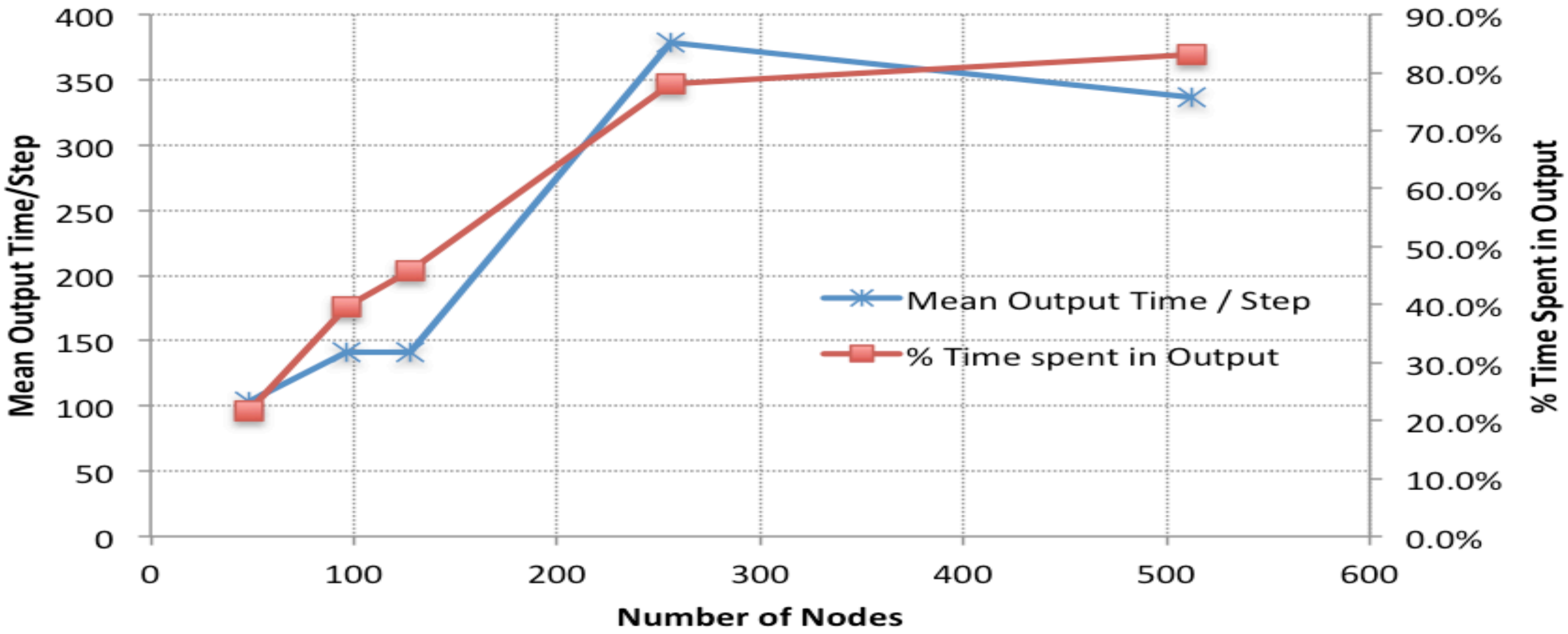
1. Serial NetCDF

- **Writing a file:**
  - All data gathered onto master MPI rank 0 using mpi_gatherv
  - Rank 0 reconstructs data array and writes to disk using serial NetCDF library
  - All other ranks block, stalling the computation, until write is complete
    - “Effective” write time as seen by compute ranks includes gather and formatting as well as actual disk write

- **Drawbacks:**
  - Easy to use, good for small rank counts, but....
  - Rank 0 requires lots memory (though can use MPMD to place it on its own node)
  - Overhead of mpi_gatherv rapidly becomes huge bottleneck at higher MPI rank counts
  - MPI/OpenMP hybrid mode can help, but eventually need another solution....

Copyright 2016 Cray Inc.

Serial NetCDF - overhead



Copyright 2016 Cray Inc.

2. Parallel NetCDF

- **Compile with parallel NetCDF enabled and set in input namelist at runtime**
- **MPI ranks aggregated into groups**
  - Number of groups = Lustre stripe count of input/output file
  - Or can be set via MPI-IO hints. For example:
    - *export MPICH_MPIIO_HINTS= "wrfout*:striping_factor=16"*
  - One aggregator from each group writes to file
  - Reduces gather time and contention
    - Cray MPI-IO is optimized to align I/O with parallel file system striping
    - Cray MPI-IO collective buffering assigns one aggregator per OST and spreads aggregators out evenly across the nodes
    - More OSTs available = more parallelism possible
- **Cray MPI-IO layer on the XC provides useful environment variables to control diagnostics.....**

Copyright 2016 Cray Inc.

MPICH_MPIIO_AGGREGATOR_PLACEMENT_DISPLAY=1

```
Aggregator Placement for wrfinput_d01
RankReorderMethod=3  AggPlacementStride=-1
 AGG    Rank       nid
 ----  ------  --------
    0       0  nid00016
    1     434  nid00041
    2     868  nid00125
    3    1302  nid00221
    4    1736  nid00245
    5    2170  nid00269
    6    2604  nid00293
    7    3038  nid00317
    8      18  nid00349
    9     452  nid00373
   10     886  nid00773
   11    1320  nid00797
   12    1754  nid00821
   13    2188  nid00845
   14    2622  nid00869
   15    3056  nid00893
```

Shows:

- How many aggregators have been assigned
- Whether rank reordering was used
- MPI rank numbers of assigned aggregators
- Node NID numbers of assigned aggregators

Note the spreading of aggregators among the MPI ranks and the nodes

Copyright 2016 Cray Inc.

MPICH_MPIIO_HINTS_DISPLAY=1

```
PE 0: MPIIO hints for wrfoutput_d01:
cb_buffer_size                        = 16777216
romio_cb_read                         = automatic
romio_cb_write                        = automatic
cb_nodes                  = 16
cb_align                  = 2
romio_no_indep_rw         = false
romio_cb_pfr                          = disable
romio_cb_fr_types         = aa
romio_cb_fr_alignment     = 1
romio_cb_ds_threshold = 0
romio_cb_alltoall         = automatic
ind_rd_buffer_size        = 4194304
ind_wr_buffer_size        = 524288
romio_ds_read             = disable
romio_ds_write            = disable
striping_factor           = 16
striping_unit             = 1048576
romio_lustre_start_iodevice = 0
direct_io        = false
aggregator_placement_stride = -1
abort_on_rw_error         = disable
cb_config_list            = *:*
romio_filesystem_type     = CRAY ADIO:
```

Look out for the value of cb_nodes – if not equal to the expected number of aggregators (stripes), we have a problem!

Copyright 2016 Cray Inc.

MPICH_MPIIO_STATS=1

```
+-----------------------------------+
| MPIIO read access patterns for
| wrfinput_d01
|   independent reads       = 1
|   collective reads        = 457452
|   independent readers     = 1
|   aggregators             = 16
|   stripe count            = 16
|   stripe size             = 1048576
|   system reads            = 7727
|   stripe sized reads      = 7512
|               total bytes for reads = 7964753971
|                                     = 7595 MiB = 7 GiB
|   ave system read size    = 1030769
|   number of read gaps     = 1
|   ave read gap size       = 1
| See "Optimizing MPI I/O on Cray XE
| Systems" S-0013-20 for explanations.
+-----------------------------------+
```

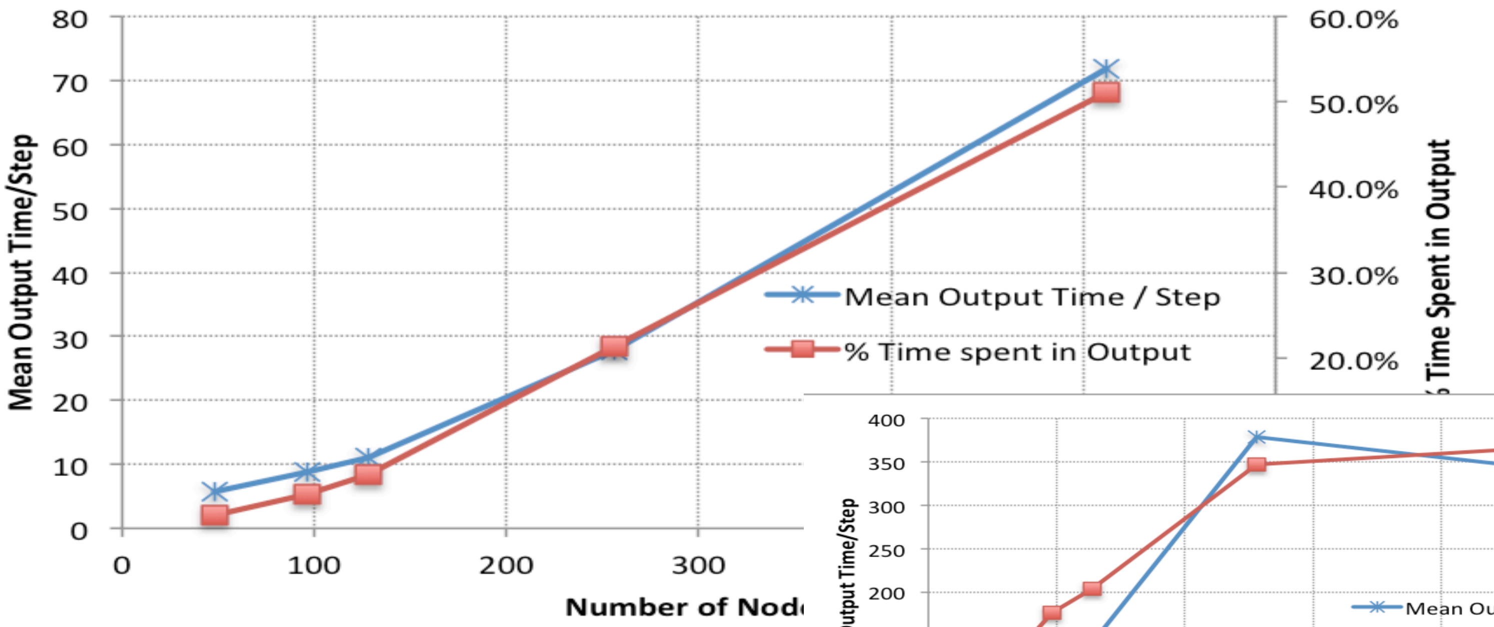
For best performance, we want many collective reads, few independent reads, and few gaps.

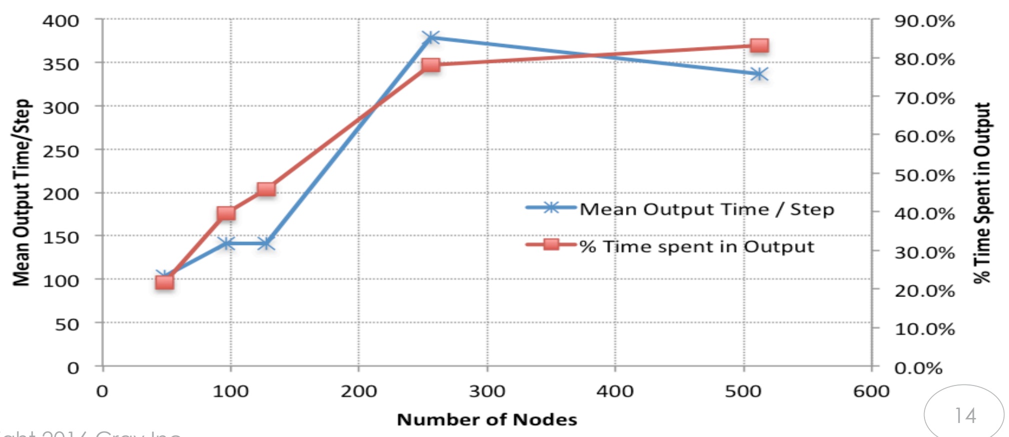
Try MPICH_MPIIO_STATS=2 for much more performance information including a timeline and data to generate bandwidth charts.

man intro_mpi

Copyright 2016 Cray Inc.

Back to WRF and Parallel NetCDF...





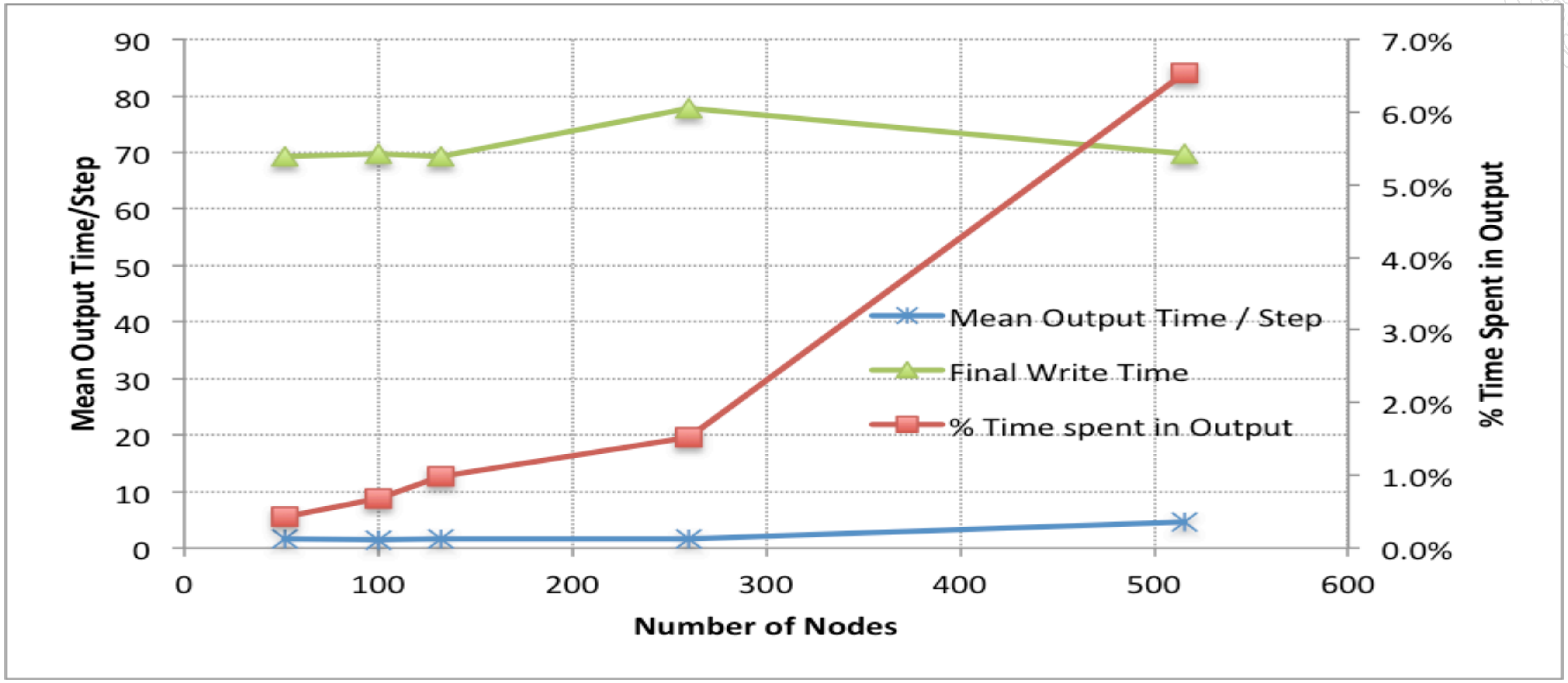
Copyright 2016 Cray Inc.

3. Quilt Servers (asynchronous I/O)

- **Quilt servers deal exclusively with I/O**
  - Groups of compute ranks are mapped onto quilt servers as evenly as possible
    - Ideally have equal numbers of ranks per server [much more important in next method]
  - Send data to assigned I/O server then continue with integration while data is formatted and written to disk asynchronously
- **Select via input namelist nio_groups of I/O servers and nio_tasks_per_group servers per group**
  - One group can only work on one output frame at a time
  - Need more than one group if write more than one frame per step (e.g. multiple domains, restart + history, etc.)
  - Need more than one group if next output step is reached before previous write is finished otherwise all ranks have to wait
- **"Effective" write time seen by compute ranks is now minimal (<1s)**
- **Actual time to write to disk much higher as performed serially by each I/O group**

Copyright 2016 Cray Inc.

Quilt Servers – Output overhead



Copyright 2016 Cray Inc.

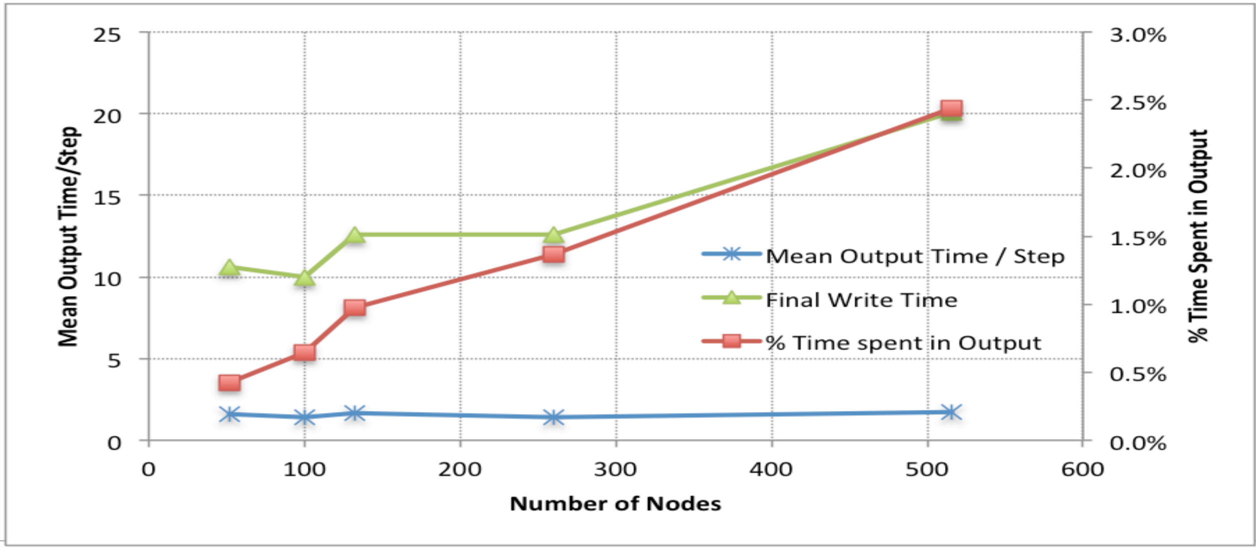
Quilt Servers: Drawbacks

- **Final write step is slow as cannot be overlapped**
  - This might not matter for a long simulation, but can overwhelm shorter forecasts or benchmarks
  - Can affect ensemble runs
  - Effect worsens at higher total rank counts
- **I/O servers require much more memory than compute ranks and so can only assign a few per node**
  - Use ALPS MPMD to achieve this (see paper)
- **Need more and more I/O groups if time between output steps decreases, e.g.,**
  - More MPI ranks so faster compute times
  - Frequent output required (e.g. severe thunderstorm forecasting)
- **... So need to assign more and more nodes to I/O**
- **What if we could speed up the output within each quilt server group?**

Copyright 2016 Cray Inc.

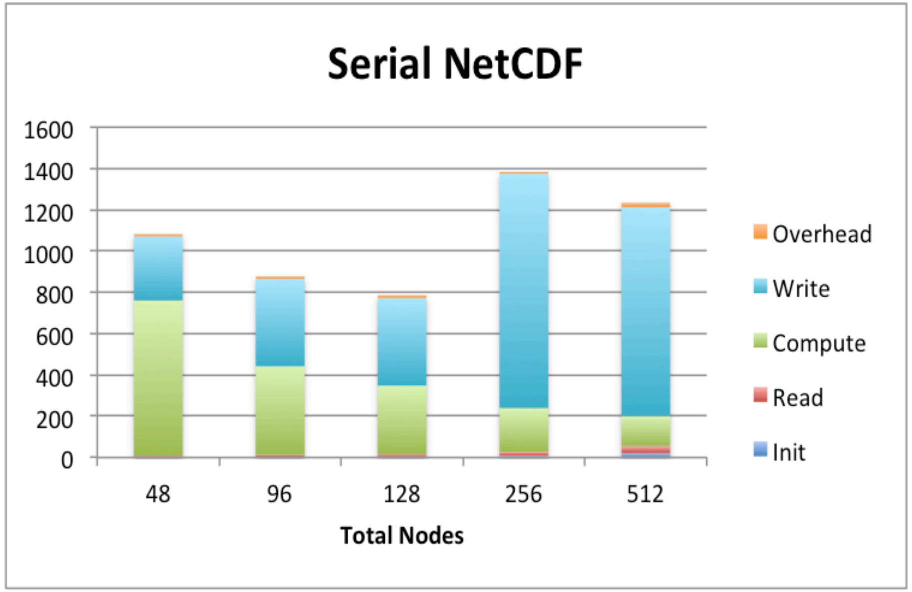
... We can! 4. Quilt Servers PLUS Parallel NetCDF

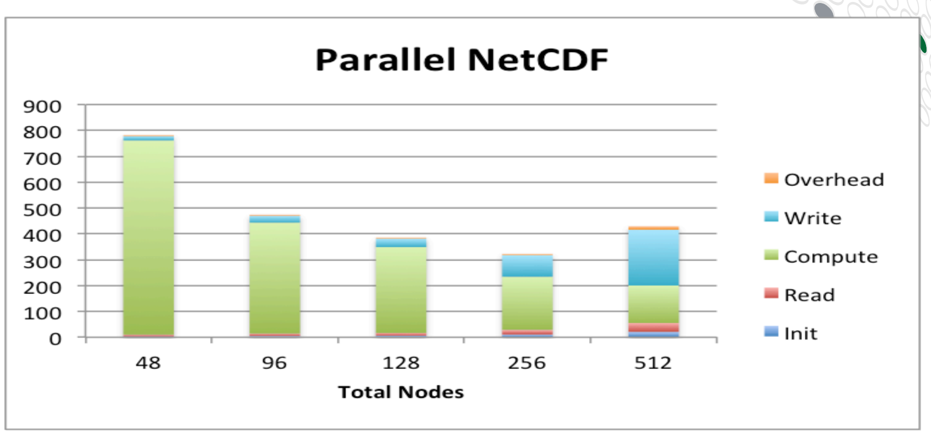
- **Percentage of time spent in output now below 3% AND final write time much lower than in previous case**
  - Wallclock time to complete forecast under half that of serial case
  - Allows more frequent output and higher scaling
  - Can handle short runs better

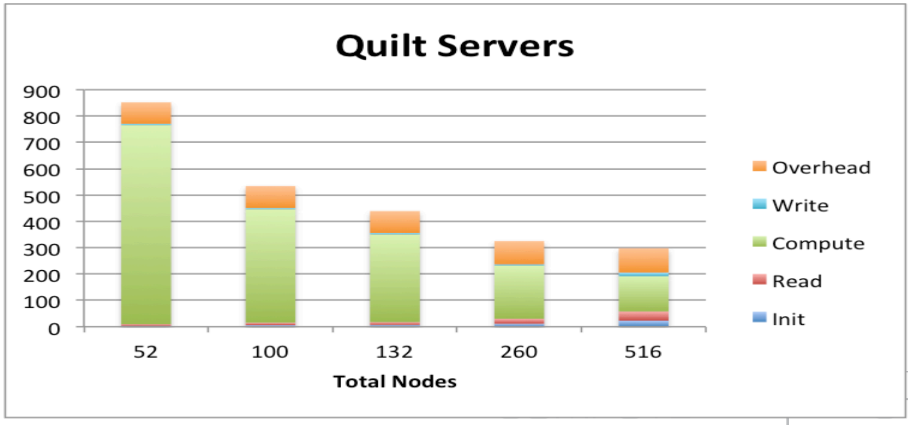


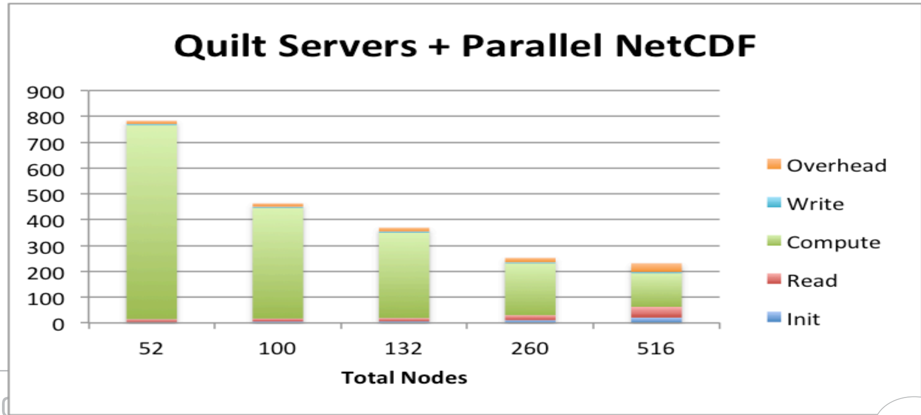
Copyright 2016 Cray Inc.

Final Comparison









Copyright 2016 Cray Inc.

Initial observations on WRF with Cray DataWarp

- **SSD-based hardware (two Intel P3608 SSD per node)**
  - Lower cost than Lustre (comparing SSU to DW), higher reliability
- **Cray-developed software, integrated with SLURM and other WLMs**
- **Input file pre-staged to DataWarp and output history files staged to permanent Lustre storage at end of run**
  - Preprocessing output or periodic restart files could be kept in scratch and not staged in or out
- **Directives to stage files are parsed by WLM (see paper for more)**

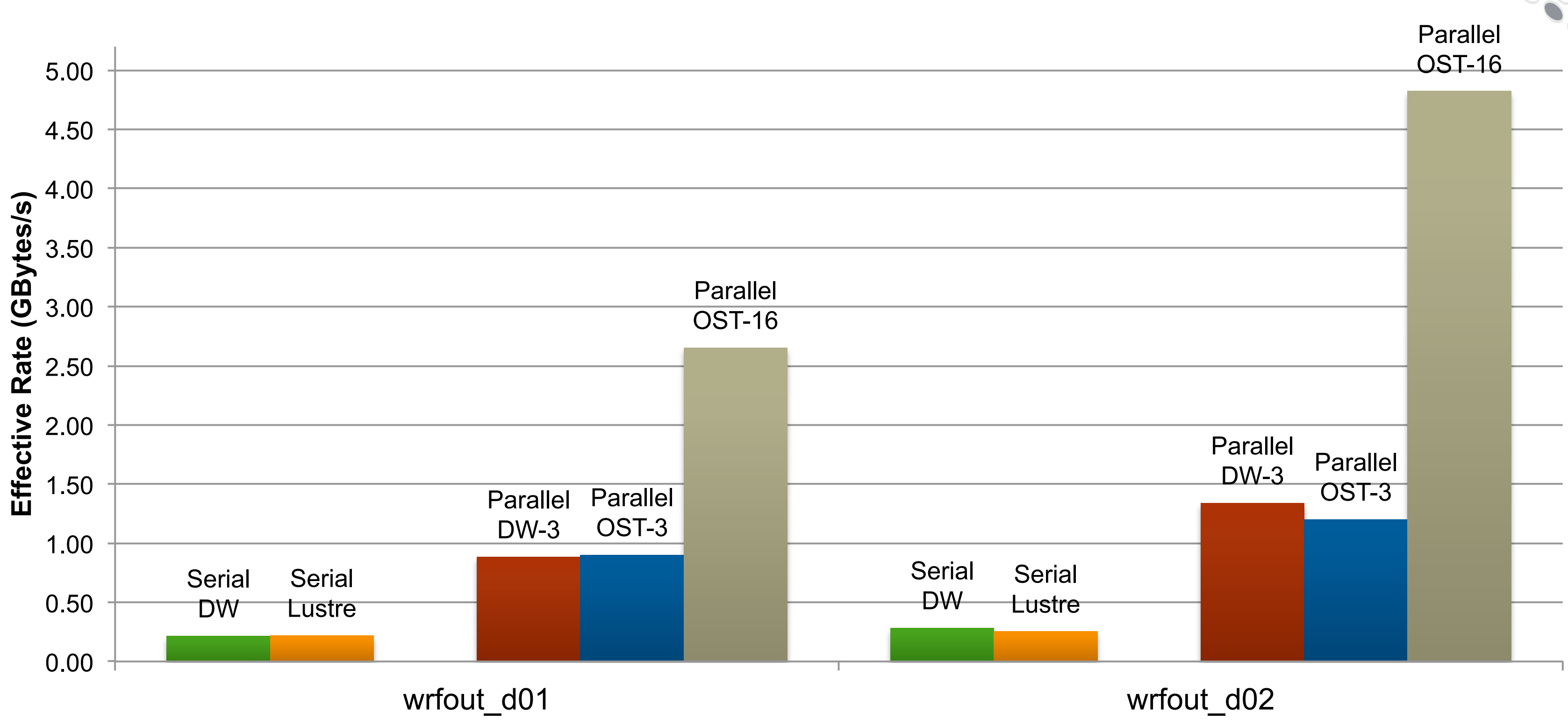
  #DW jobdw type=scratch access_mode=striped capacity=1150GiB

  #DW stage_in type=file source=INPUT/wrfinput_d01 destination=$DW_JOB_STRIPED/wrfinput_d01

  #DW stage_out type=file destination=OUTPUT/wrfout_d01_2015-03-10_00_00_00 source=$DW_JOB_STRIPED/wrfout_d01_2015-03-10_00_00_00
- **Compared 3 DataWarp nodes against 3 and 16 Lustre OST stripes**
  - Ongoing studies at KAUST scaling up to 100 nodes and beyond

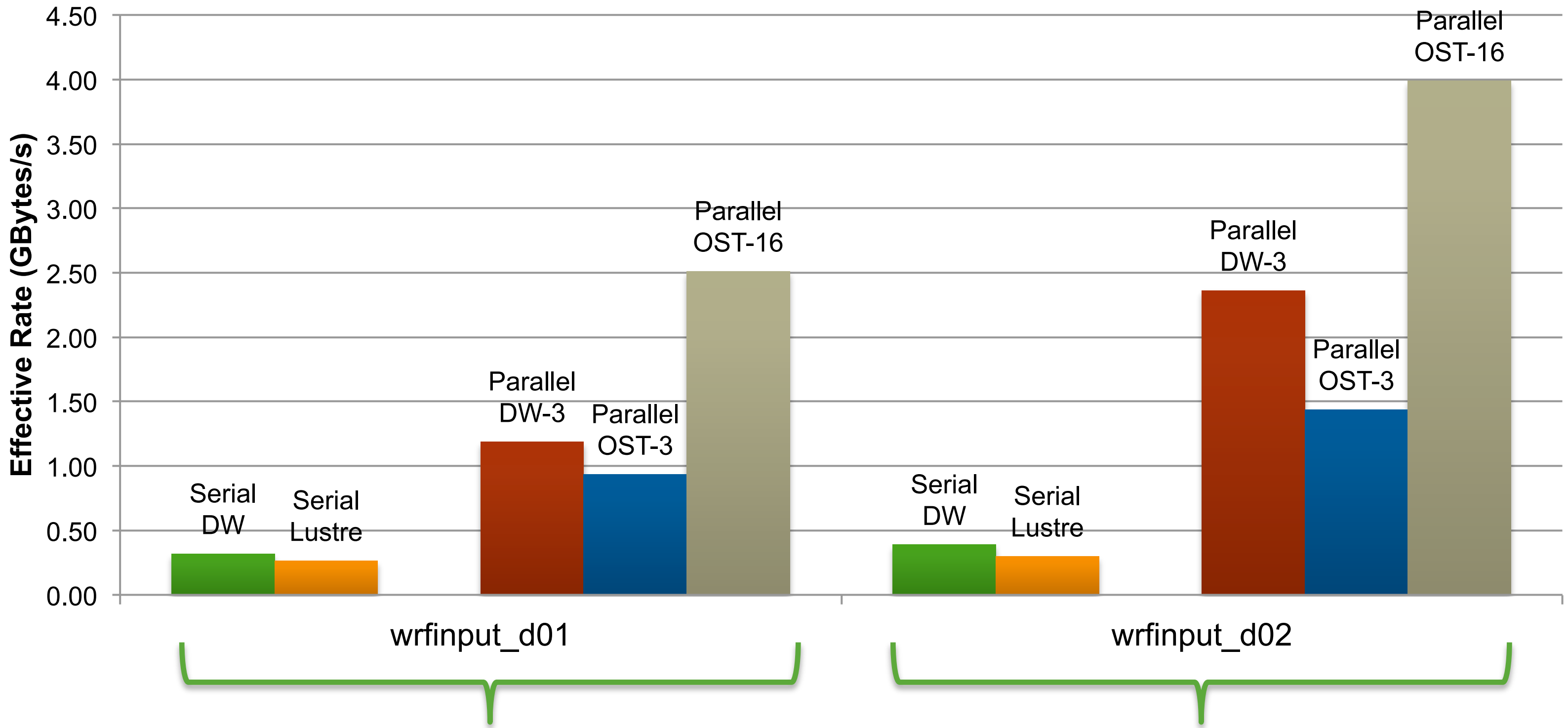
Copyright 2016 Cray Inc.

DataWarp/Lustre Comparison - Output



Copyright 2016 Cray Inc.

DataWarp/Lustre Comparison - Input



Copyright 2016 Cray Inc.

Summary

- **I/O overhead can drastically limit scaling of WRF to higher core counts and higher output frequencies**
- **Existing methods, especially parallel NetCDF + quilts can effectively hide much of the I/O overhead and enable realistic scaling**
  - Cray MPI-IO layer a great advantage on XC
- **Use of Parallel NetCDF + quilts can improve time to forecast compared to serial NetCDF by over 2x**
  - Method should be more widely used! See paper for hints on usage.
- **Cray DataWarp could be a great option for WRF I/O**
  - Not only for forecast input and output, but as scratch storage during pre and post processing over an entire workflow

Copyright 2016 Cray Inc.

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 publically 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, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM, 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 following 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 in this document are the property of their respective owners.

Copyright 2016 Cray Inc.

Q&A

Tricia Balle
pburgess@cray.com

Pete Johnsen
pjj@cray.com

Copyright 2016 Cray Inc.