

Global Arrays Runtime

Presenter: Daniel Chavarría-Miranda

Research & technical leads: **Abhinav Vishnu**, Jeff Daily,
Sriram Krishnamoorthy, Nathan Tallent

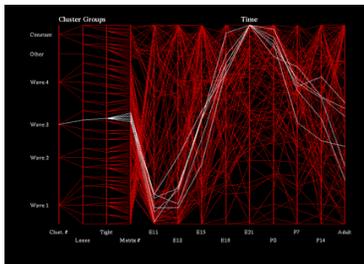


Pacific Northwest
NATIONAL LABORATORY

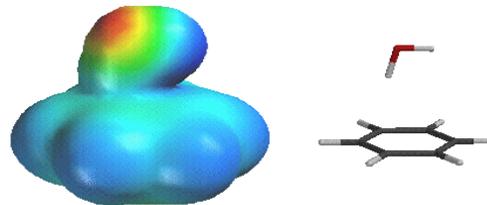
Proudly Operated by Battelle Since 1965

Global Arrays: Global-view Distributed Arrays

- ▶ Library-based, compiler-free PGAS programming model (20+ years of R&D)
- ▶ Deployed on largest systems (Mira@ALCF, Titan@OLCF, Edison@NERSC)
 - Available at 3,000+ sites around the world
- ▶ Large group of applications:
 - **Chemistry:** NWChem (**223K cores**), Columbus, GAMESS-UK, Molpro, Molcas, CP2K
 - **Subsurface:** STOMP (**130K cores**), Paraflow
 - **Bioinformatics:** ScalaBLAST (**18K cores**), Pgraph (**120K cores**), ArrayDB
 - **CFD:** Tethys
 - **Astrophysics:** Chimera
 - **Power Grid:** MCA, Kalman-filter
 - **Quantum Monte Carlo:** GAEinspline
 - **Geoscience:** PAGODA



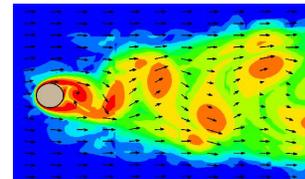
Bioinformatics:
ScalaBLAST,
18K procs.



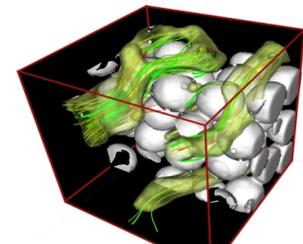
electronic structure
chemistry

GA is the standard programming
model

NWChem, 223K procs.



Subsurface
transport:
STOMP, 130K procs.



GA Runtime Features & Design

- ▶ Original GA runtime: Aggregate Remote Memory Copy Interface (ARMCI) [IJHPCA'06]
- ▶ Supports one-sided RMA operations: get, put, accumulate
 - Over native network libraries [IPDPS'04, IPDPS'07]
- ▶ Principal design criteria:
 - True one-sided communication: synchronous & asynchronous
 - Active communication progress agent [SC'06]
 - Optimized communication patterns for global array slice access
 - e.g. multidimensional strided [ICPP'08]
- ▶ Underlying execution model: Bulk Synchronous Parallelism (BSP)
 - Realized by UNIX processes + communication server threads [CF'06]

GA Runtime Features & Design (cont.)

- ▶ New runtime: Communication subsystem for Exascale (ComEx)
 - More flexible code base
 - Support for native network libraries [HiPC'12]
 - Support for different MPI-based transfer layers [HiPC'14]
 - Improved memory & resource usage for scalability [CCGrid'13]
 - Support for fault resilience [HiPC'10, CLUSTER'09, EuroPar'11, PDP'11, JCTC'11, JCTC,13]
 - Support for reducing energy consumption [JSC'13, GreenComm'10]
 - Diagnosing network contention [PPoPP'15]
- ▶ Beyond the process-based BSP execution model
 - TASCEL (TASk SchEduling Library): support for distributed task-based work stealing at scale [SC'09, ICPP'12, HPDC'12, ICS'12]
 - Global Futures: data-centric active messages [CCGrid'12]

Diagnosing Network Contention



- ▶ Growing ratio of cores to network bandwidth
- ▶ Growing interest in PGAS programming models
- ▶ PGAS models use one-sided messages

- Examples:

`x: get *y`

`x: nb_get *y`

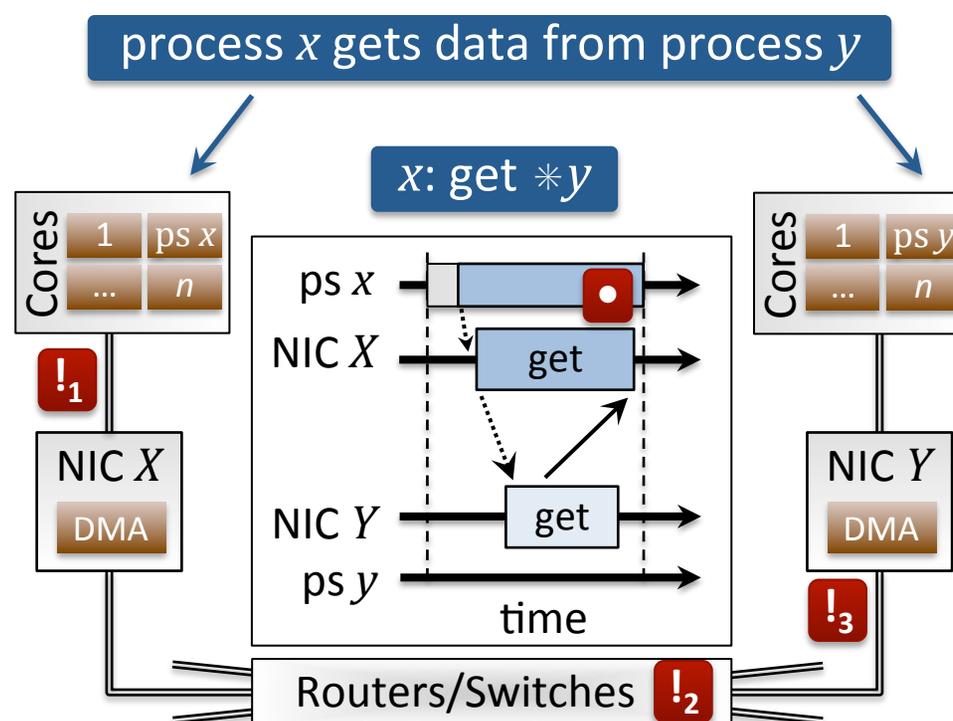
`x: put *y`

`x: nb_put *y`

- Readily implemented on RDMA hardware (remote direct memory access)

- ▶ Contention (!_x) can increase message blocking time (•) by integer factors

- Cf. two-sided: blocking time function of contention & sync



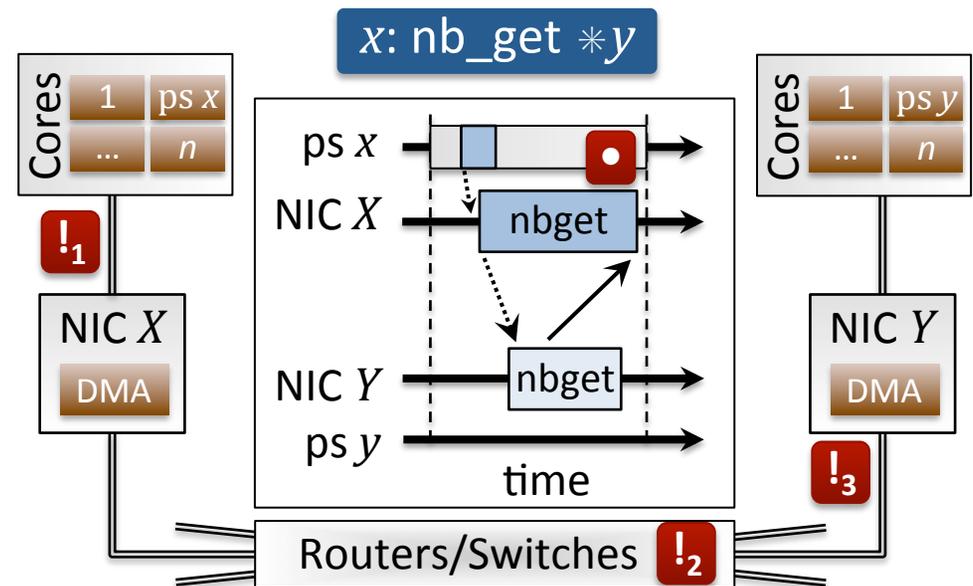
Difficult to Reason About Network Contention



- ▶ Asynchrony may lower contention
 - Avoid bursts of communication during collectives
- ▶ Asynchrony may increase contention
 - Introduce network hot spots writing (putting, scattering) or reading (getting, gathering) data

▶ Challenges:

- Detect: Contention at network interior (!₂) and message target (!₃) is remote from application execution (!₁)
- Attribute: Message delivery asynchronous with cores (•)



Detect one-sided network contention, quantify severity, attribute to app

Diagnosing Contention in NWChem DFT Carbon240



PIC: QDR InfiniBand (oversubscribed fat tree, static routing); task 46; 64 nodes

NWChem:

- asynchronous
- tasks compute on distributed arrays
- avoids collectives

source code

```

87  call fock_get
88  $   llo, lhi, klo, khi, dlk, tmp)
89  call fock_get_blk(nfock, vg_dens,
90  $   llo, lhi, ilo, ihi, dli, tmp)
91  call fock_get_blk(nfock, vg_dens,
92  $   llo, lhi, jlo, jhi, dlj, tmp)
    
```

100% of contention – 65% of execution time – due to get

Calling Context View | Callers View | Flat View

← call chain

	t_{blocked}	$t_{\text{contention}}$ (Sum/E)	$t_{\text{contention-initiator}}$	$t_{\text{contention-target}}$
1167: ga_get_	4.78e+11 99.9%	4.76e+11 99.6%	2.32e+09 84.5%	4.74e+11 99.7%
56: fock_get_blk	4.78e+11 99.8%	4.76e+11 99.6%	2.22e+09 80.9%	4.74e+11 99.7%
89: fock_2e_cache_dens_fock	3.70e+11 77.3%	3.69e+11 77.2%	1.32e+09 48.2%	3.68e+11 77.4%
91: fock_2e_cache_dens_fock	3.61e+10 7.5%	3.58e+10 7.5%	2.81e+08 10.2%	3.55e+10 7.5%
62: fock_2e_cache_dens_fock	2.63e+10 5.5%	2.61e+10 5.5%	2.37e+08 8.4%	2.60e+10 5.5%
87: fock_2e_cache_dens_fock	1.68e+10 3.5%	1.67e+10 3.5%	1.30e+08 4.6%	1.66e+10 3.5%
71: fock_2e_cache_dens_fock	1.49e+10 3.1%	1.48e+10 3.1%	1.30e+08 4.6%	1.47e+10 3.1%
80: fock_2e_cache_dens_fock	1.25e+10 2.6%	1.25e+10 2.6%	1.22e+08 4.3%	1.24e+10 2.6%

Contention mostly at get's target (location of data)

- 4.7e11 vs. 2.2e9 (μs)

Moving hot-spot (at target) reading data

- 2 gets (read) for each acc (write)
- gets contend for out links sending to initiator
- gets go over different links than accs

Change global array distribution to improve total runtime by 20%.

TASCEL: Task-Parallel Abstractions

- ▶ **Finer specification of concurrency, data locality, and dependences**
 - Convey more application information to runtime
- ▶ **Adaptive runtime system to manage tasks**
- ▶ **Application writer specifies the computation**
 - Writes optimizable code
- ▶ **Tools to transform code to efficient implementation separated from the application**



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Work Stealing

- ▶ A worker begins with one/few tasks
 - Tasks spawn more tasks
 - When a worker is out of tasks, it steals from another worker

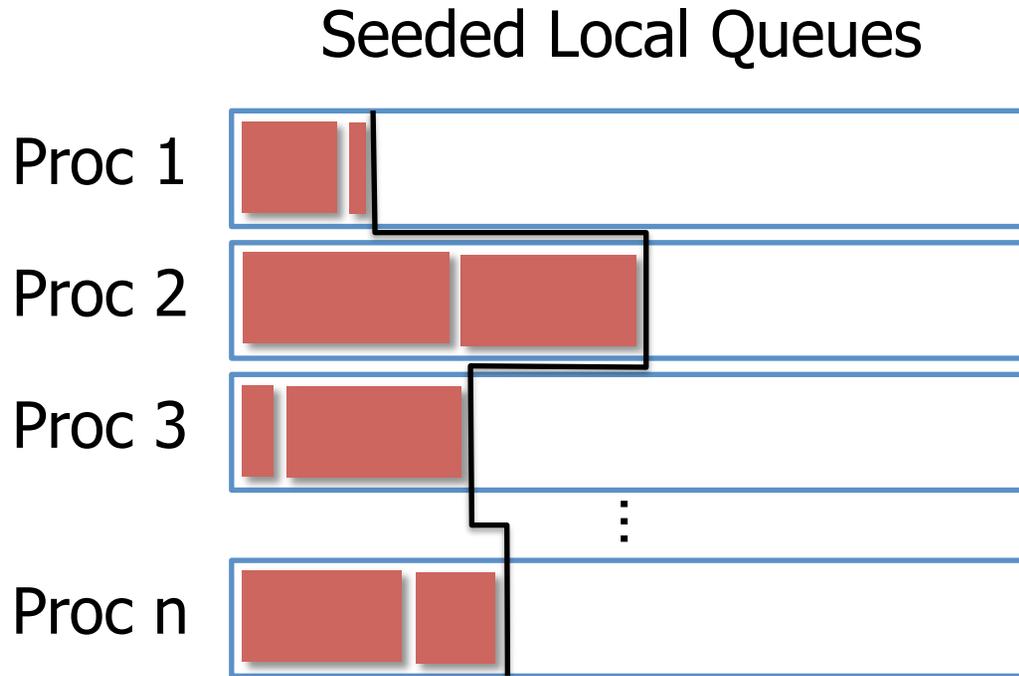
- ▶ A popular task scheduling strategy
 - Well-studied load balancing strategy
 - Provably efficient scheduling
 - Understandable space and time bounds



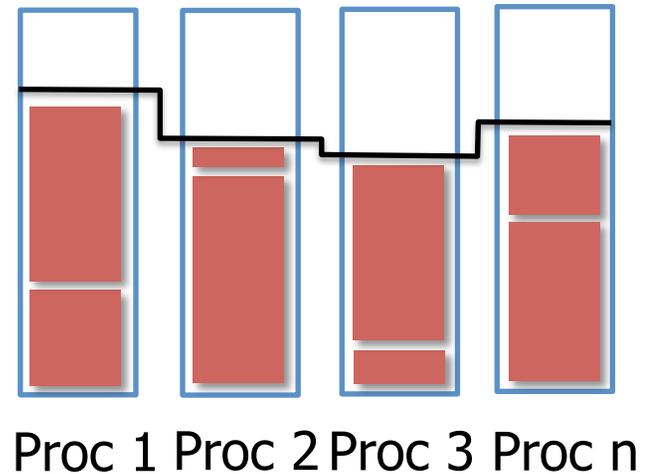
Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Retentive Work Stealing



Actual Executed Tasks



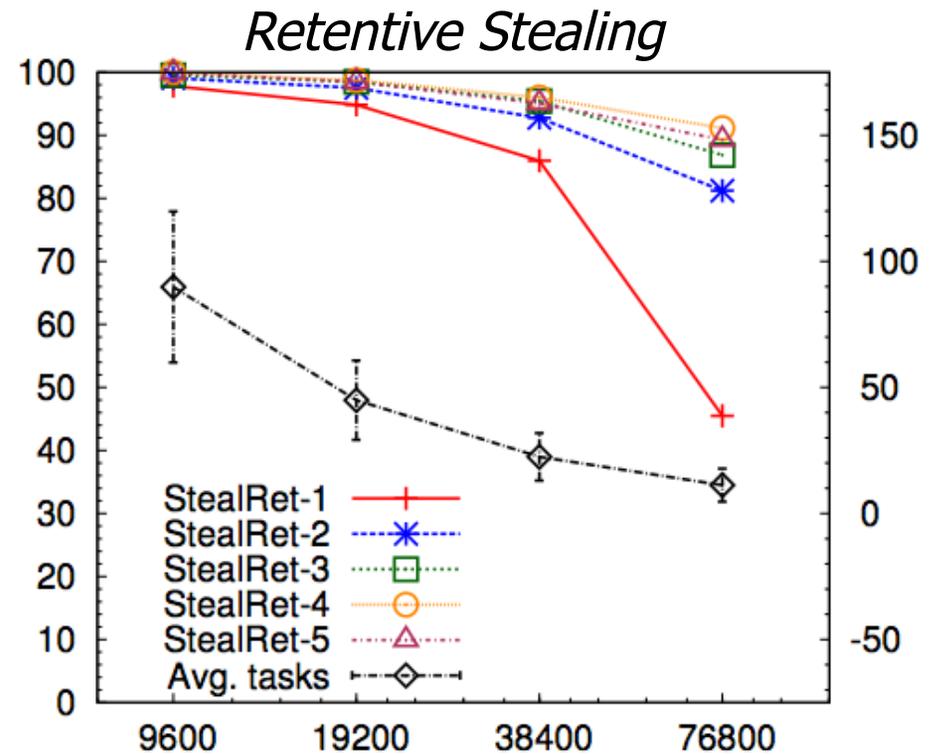
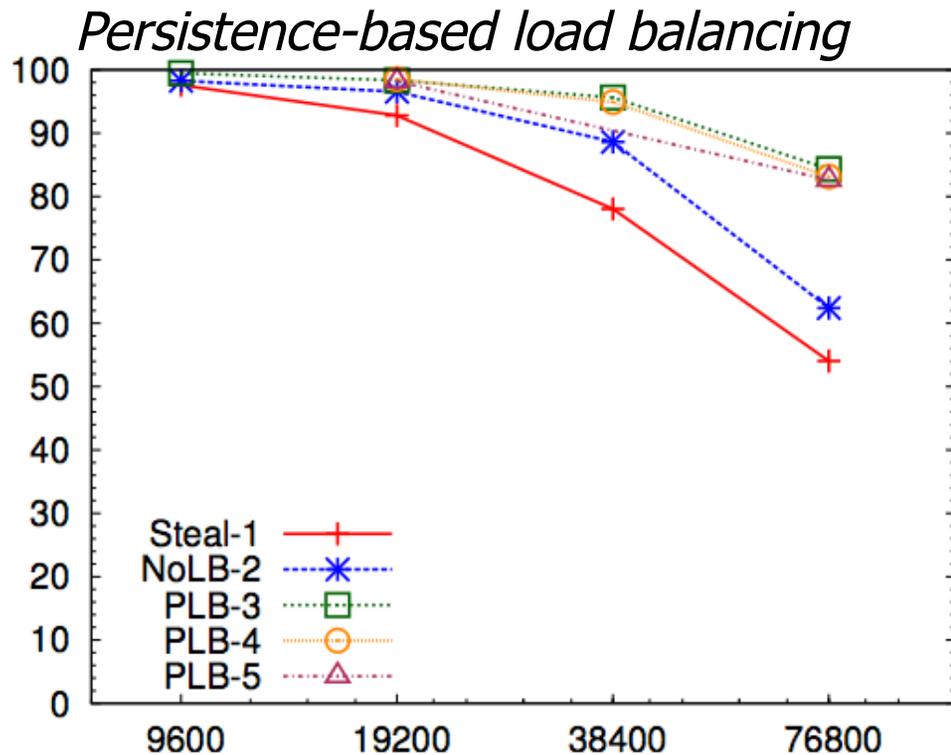
Intuition: Stealing indicates poor initial balance



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

Hopper: Performance



- ▶ Persistence-based load balancing “converges” faster
- ▶ Retentive stealing also improves efficiency
- ▶ Stealing effective even with limited parallelism

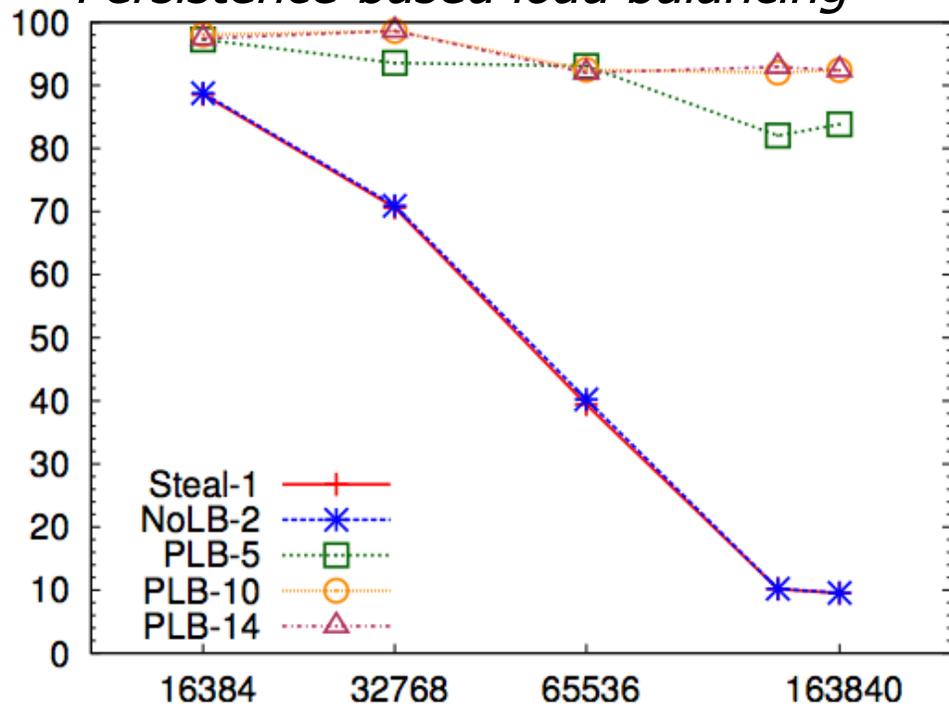


Pacific Northwest
NATIONAL LABORATORY

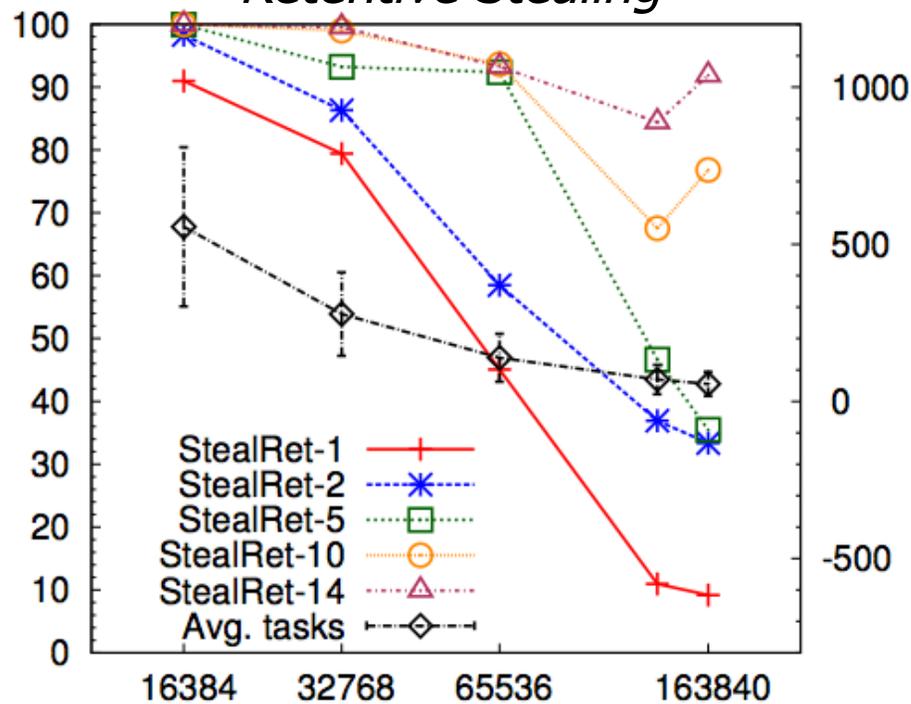
Proudly Operated by Battelle Since 1965

Intrepid: Performance

Persistence-based load balancing



Retentive Stealing



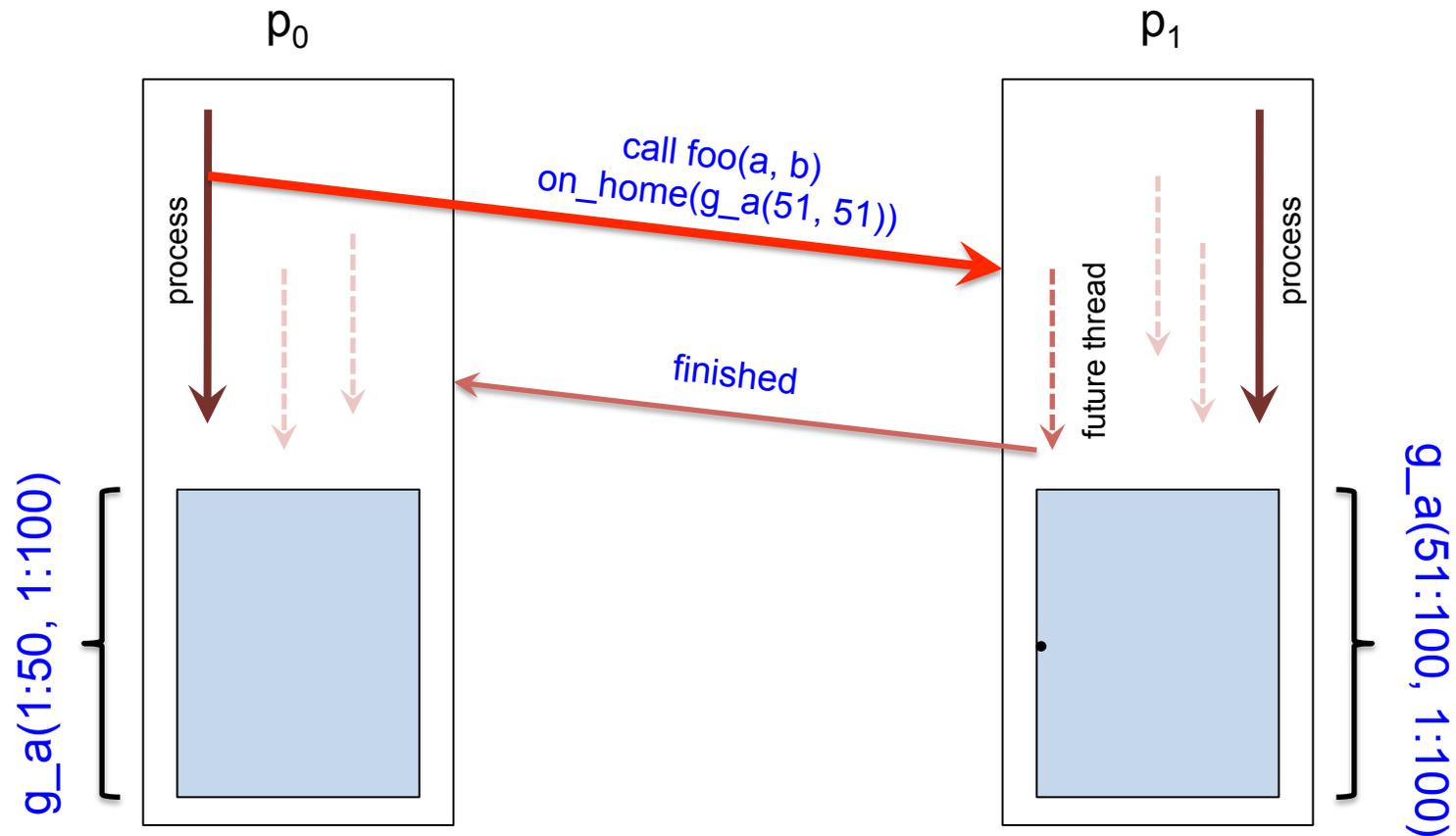
- ▶ Worse performance for the first iteration
- ▶ Converges to a better efficiency than on Hopper



Pacific Northwest
NATIONAL LABORATORY

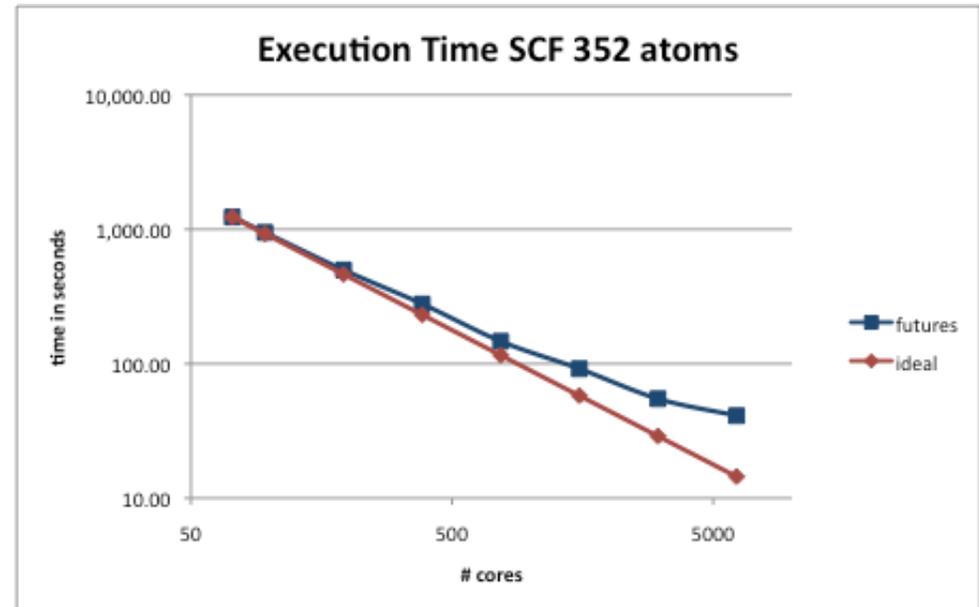
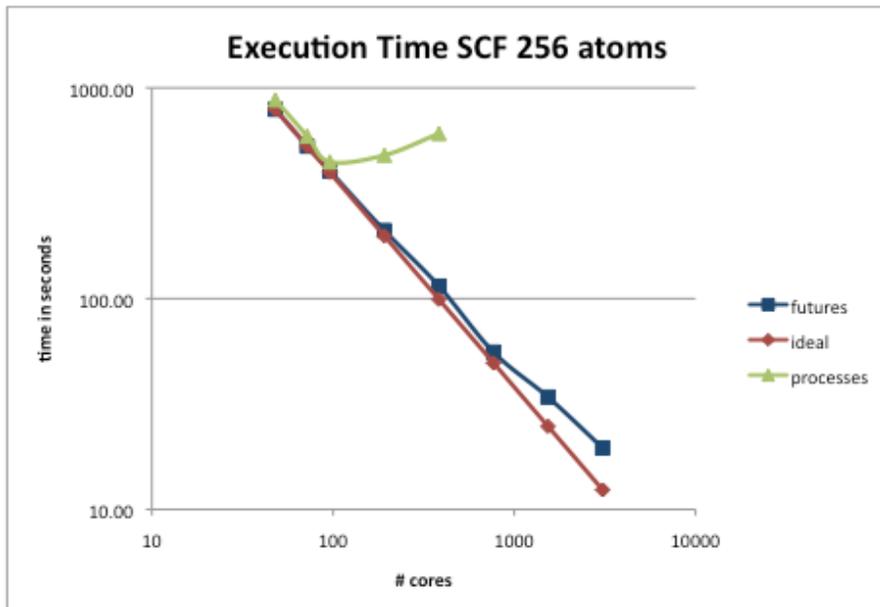
Proudly Operated by Battelle Since 1965

Global Futures – moving computation to data



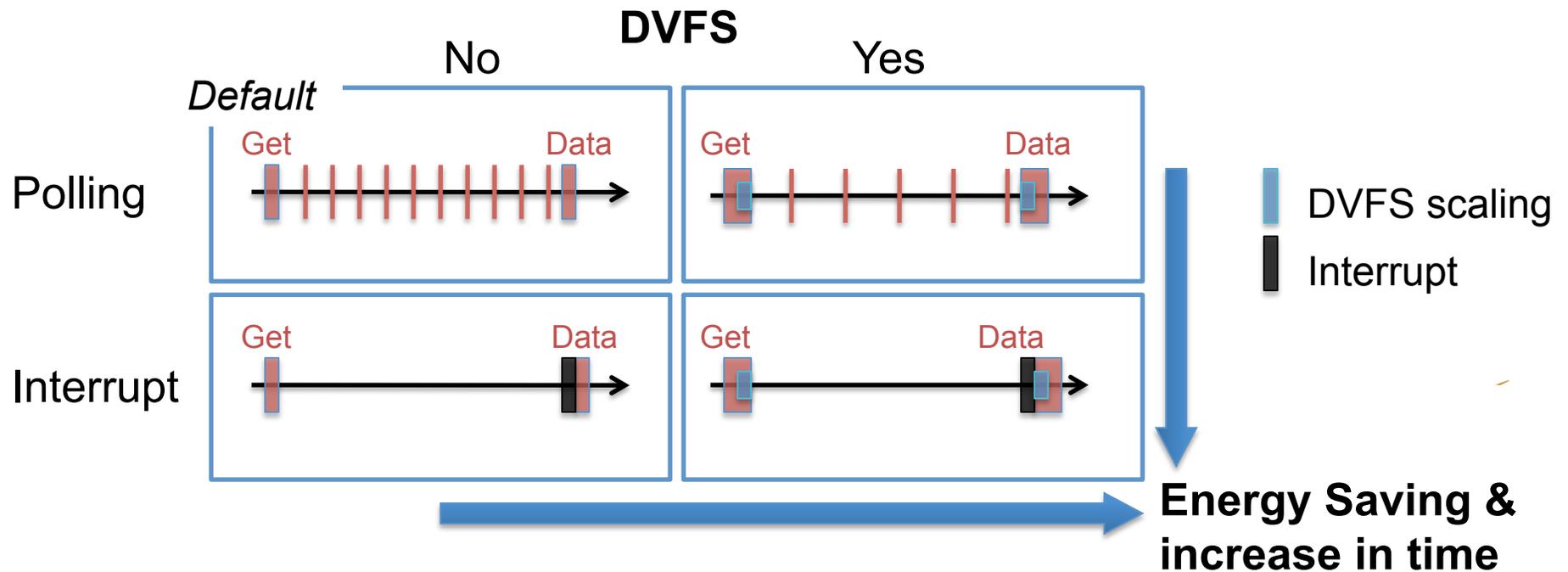
Global Futures

- ▶ Chinook (2310 node DDR Infiniband cluster)
 - 1 GA process per node, 6 threads
 - 256 atom system with tiles of size 40
 - 352 atom system with tiles of size 40
- ▶ Caching remote data improves scalability
 - Including accumulates
 - Data reuse across threads



Energy Efficiency Mechanisms for Communications

- ▶ **Dynamic Voltage/Frequency Scaling (DVFS)**
 - Scale down during communication, Freq (core), Voltage (socket)
 - But may lead to increased latency
- ▶ **Interrupt Driven Execution**
 - Yield CPU & wake up on network event, BUT may increase latency



Handling Different Data Transfer Types Leads to Differences in Possible Gains

- ▶ Contiguous data transfer
 - Request DVFS scale down after data transfer request initiated
 - DVFS scale up after data transfer has been completed
- ▶ Non-contiguous (e.g. strided) data transfer
 - Requires data copy to/from intermediate buffers
 - DVFS/Interrupt based execution performed after data transfer request initiation
- ▶ Less potential gain on non-contiguous transfers



Summary

- ▶ Experience with GA, ARMCI & ComEx is very valuable for the community
 - Real production applications running on it at scale
 - Major users of LCFs and NERSC resources
 - Have addressed key issues related to scaling, performance, portability, fault tolerance
 - Insights & inroads into emerging obstacles
 - Energy & power
 - Resilience
- ▶ Higher-level abstractions can be supported without compromising performance
 - Global-view access, task-based execution
- ▶ Flexible runtime framework can support various execution styles
 - Task-based, work-stealing
 - Data-centric active messages



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

**THANK YOU
QUESTIONS?**



Proudly Operated by Battelle Since 1965