SAND2019-5810C

Optimizing a New DARMA Runtime for Load Balancing EMPIRE

Presented by: Jonathan Lifflander (PI), SNL

Current Team:

Ulrich Hetmaniuk (NGA) Phil Miller (NGA)
Nicolas Morales (SNL) Philippe P. Pébaÿ (NGA)
Meriadeg Perrinel (NGA) Hoby Rakotoarivelo (NGA)
Nicole Slattengren (SNL) Gary Templet (SNL)

NGA = NexGen Analytics, **SNL** = Sandia National Labs

Charm++ Workshop 2019, May 1-2nd

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

What is DARMA?

Sandia National Laboratories

- DARMA provides C++ abstractions for asynchronous many-task (AMT) programming models
  - Driven directly by Sandia's application needs
  - Was previously part of Sandia's ATDM (Advanced Technology, Development, and Mitigation) program, building a technical roadmap for next-generation code development
  - Recently migrated to CSSE (Computational Systems & Software Environment)
- History
  - Many years ago it was a resilience project
  - In 2015, surveyed and evaluated existing AMT runtimes
  - In 2016-2017, built C++ abstractions to bridge AMT models
    - Focused on sequential semantics and advanced PM abstractions

What is DARMA?

Sandia National Laboratories

- In 2018-now:
  - Focused on producing AMT runtime software for EMPIRE, a full-fledged Sandia application
    - Building flexible PM abstractions that interoperate with MPI
    - Researching distributed load balancing strategies for large-scale deployment
      - Philippe Pébaÿ will talk about ongoing LB research after this
  - Less emphasis on bridging the gaps across existing AMT runtimes
  - More emphasis on building production quality software for AMT
  - DARMA team works closely with the EMPIRE team
    - EMPIRE development used as a catalyst to drive programming model research
    - Goal: to reimplement parts of EMPIRE with DARMA to reap benefits from load balancing

vt (Virtual Transport)

Sandia National Laboratories

- In 2017, VT was originally initiated as a “backend”
  - Started as a lower-level runtime
  - Provides maximum MPI interoperability
  - Does not strictly follow a specific PM (PERL mentality)
- Design requirements
  - Abstracts data movement
  - Provides easy, thin endpoint virtualization
  - Asynchronous, distributed event completion & coordination
  - Thin layer on top of MPI
    - Has some analogous constructs to MPI
  - Written entirely in C++: mostly C++11 (with a little C++14)
    - Compiles with gcc 4.9, clang 3.9, Intel 18
  - Open-source license (on Github, will be public soon)

vt (Virtual Transport)

Sandia National Laboratories

- Control abstractions
  - A “node” -> MPI rank
    - While it can spawn threads, the primary use case is with Kokkos kernels
  - A “PendingSend” -> the root of a task subtree
    - An operation that you may use to sequence wrt another action
  - A “group” -> MPI group
    - A subset of nodes with a associated communicator VT builds for you
    - Can be constructed collectively or rooted
  - A “virtual context” -> a migratable C++ object providing handler state
    - Similar to a Charm++ chare with different semantics
  - A “virtual context collection” -> a set of virtual contexts
    - Similar to a Charm++ chare array
  - An “epoch” -> a collective/rooted group of operations
    - Perform termination over a subset of control/data operations

vt (Virtual Transport)

Sandia National Laboratories

- Data abstractions
  - An “active message” -> a C++ POD object
    - May be serialized or not
    - May be sent to a handler or destination-oblivious endpoint
  - RMA (Remote Memory Access)
    - Directly access memory owned by another task or node
      - Reduces copying and overdecomposition overheads
    - May get/put data directly to a node or virtual context
    - May or may not use hardware primitives (depending on type)
      - Subset of functionality is currently backed by MPI one-sided
    - Has windows similar to MPI which can be linked to an epoch
- Load balancing
  - HierarchicalLB from HPDC’12 paper: *Work Stealing and Persistence-based Load Balancers for Iterative Overdecomposed Applications*
    - Jonathan Lifflander, Sriram Krishnamoorthy, Laxmikant V. Kale

vt (Virtual Transport)

Sandia National Laboratories

- Over 2 years of development
  - 55k lines of code (sloccount)
  - >600 unit + integration tests
  - Successful mini-app runs on 100k+ cores of Trinity
- Software engineering
  - The team as grown significantly that works on VT
  - We are working on improving software engineering processes
    - Code reviews, automated testing
- Has a combination of research/experimental and production components

EMPIRE Physics Application

Sandia National Laboratories

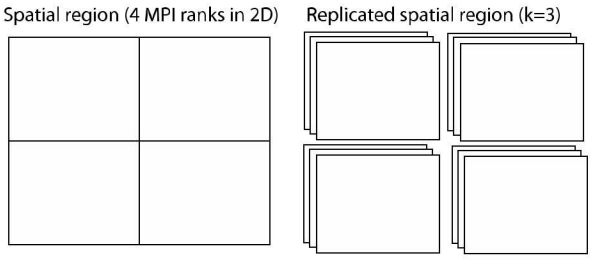
- Electromagnetic/electrostatic plasma physics application
  - Encompasses particle-in-cell (PIC), fluid, and hybrid
  - Uses Trilinos for solvers and STK (Sierra Toolkit) for meshing
  - Part of Sandia's ATDM program
    - Goal: build the next-generation apps that will replace the current production ones
    - Expected to replace current production capability in ~3-5 years as it is hardened, verified, and validated
  - Currently, the DARMA team is focused on a part of EMPIRE-PIC
    - Met a hard deadline in FY19-Q1 to demonstrate the feasibility of the DARMA software stack to improve performance through load balancing
    - Built a proof-of-concept overhaul of the PIC portion in EMPIRE in Q1
    - Currently finishing the full DARMA implementation of the EMPIRE-PIC code on the main branch of EMPIRE

EMPIRE-PIC: MPI iteration structure

Sandia National Laboratories

- In MPI, each rank has:
  - A single piece of the mesh, decomposed *a priori*
  - The set of particles that reside in that spatial region
- Each PIC iteration (at a high level):
  - (1) Perform all the "pre-move" particle-related operations
    - Particle injection, weight fields (associated with mesh)
  - (2) Execute the PIC move kernel
    - Particles traverse elements of the local mesh block
    - Send/receive any outgoing/incoming particles at local mesh boundaries
  - (3) MPI all-reduce counting particles received
    - If count is non-zero, go to (2); else PIC move is complete go to (4)
  - (4) Perform post-move operations
    - Sort particles, update counts, weight charge
  - (5) Run the solver on the fields (in Trilinos)

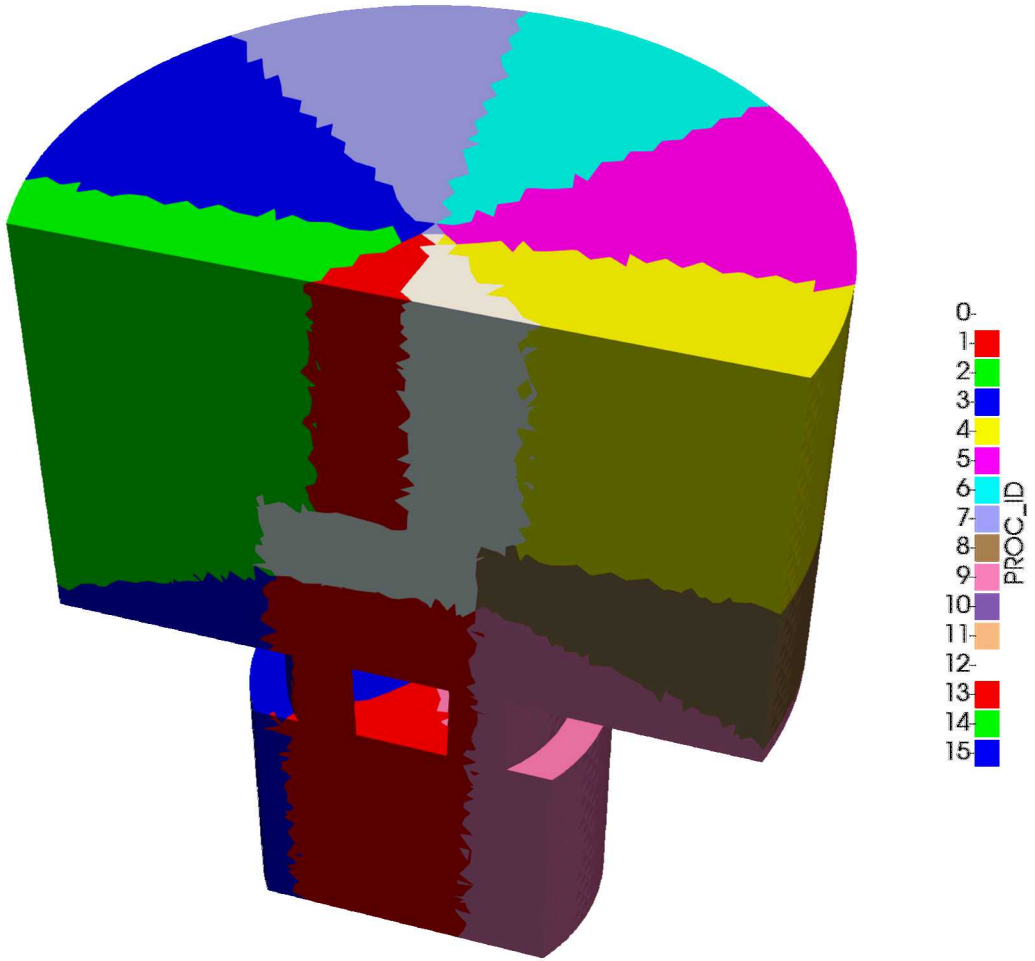
EMPIRE-PIC Tasking Challenges



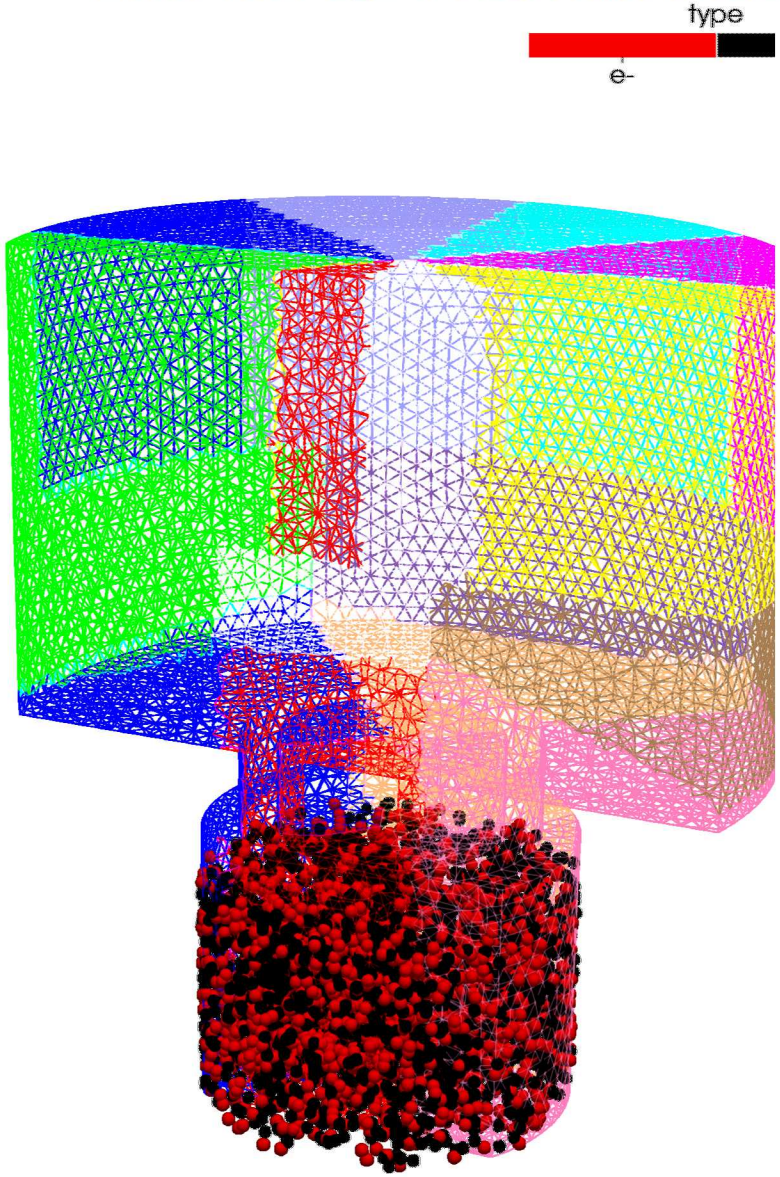
Sandia National Laboratories

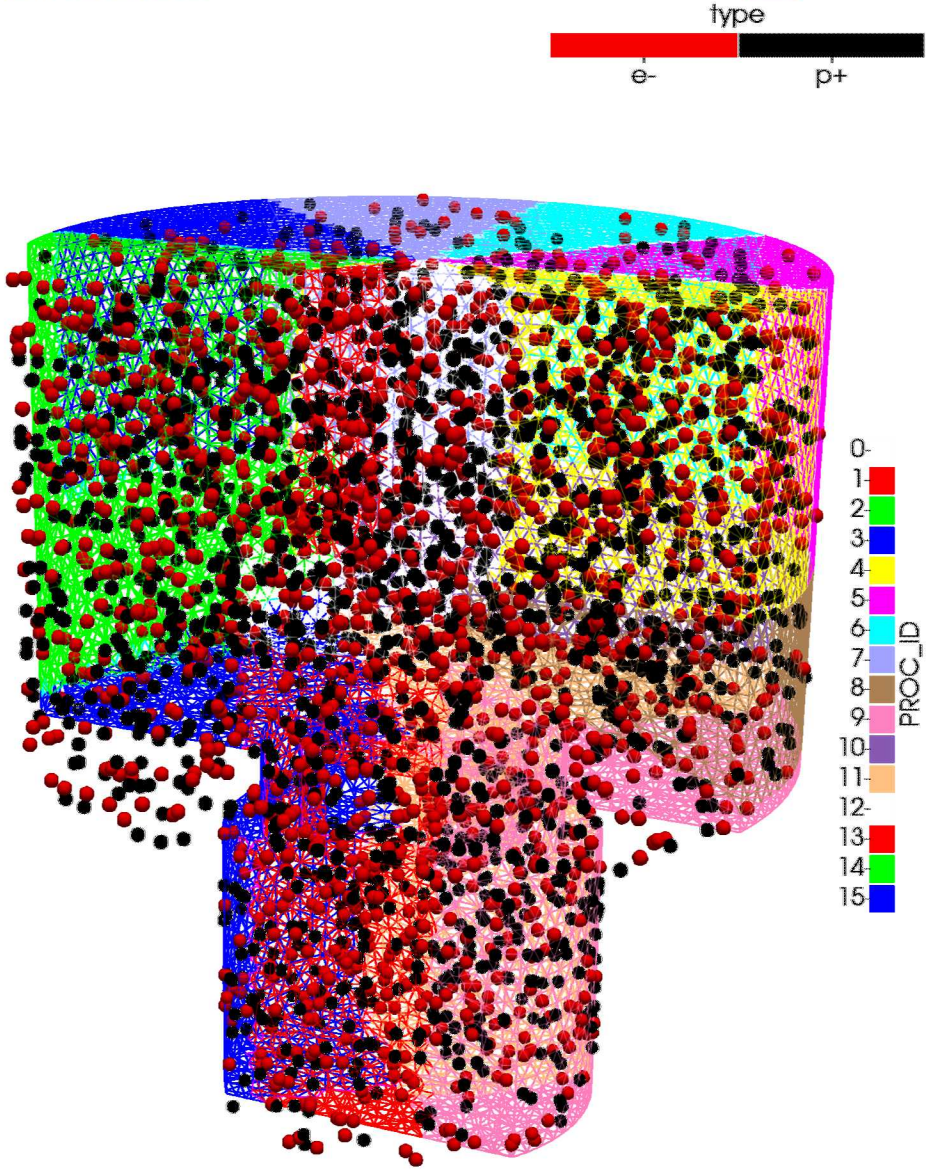
- Overdecomposing existing mesh blocks in EMPIRE is difficult
  - Mesh may be unstructured
  - Baseline mesh decomposition is fixed
    - Solver (Trilinos) and other parts of the code will not change
  - Sierra Toolkit handles mesh decomp and ghosting
- In FY19-Q1, replicate mesh blocks to overdecompose
  - Each MPI rank creates *k* replicants that operate across the same spatial region
  - VT collection uses a 2D indexing scheme to identify elements: (rank,*k*)
  - When a move pushes a particle to a neighbor, pick any replicant for the appropriate region (rank,0..k-1) to invoke the kernel
  - Each iteration the replicants are updated with new solver data
- Longer term: 2-level mesh partitioning scheme is needed

EMPIRE Q1 Problem Mesh (16 ranks)

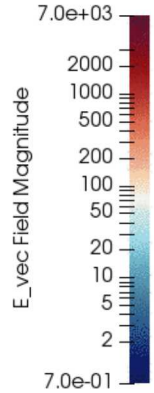


EMPIRE Q1 Problem Mesh & Particles





EMPIRE Q1 Problem Mesh & Particles Animation



Timestep : 0
Simulation Time : 0

EMPIRE-PIC Tasking Challenges

Sandia National Laboratories

- PIC is composed of a partially ordered sequence of operations performed varies depending on the physics
  - Each task/element in the collection does not know the sequence or ordering constraints
  - Operations and ordering constraints vary across elements
- After LB runs, we want operations to be dispatched locally (unless there is data exchange)
  - Instead of broadcasting/point-to-points to start a given operation on all collection elements, loop over local elements residing on a node

EMPIRE-PIC Tasking Challenges

Sandia National Laboratories

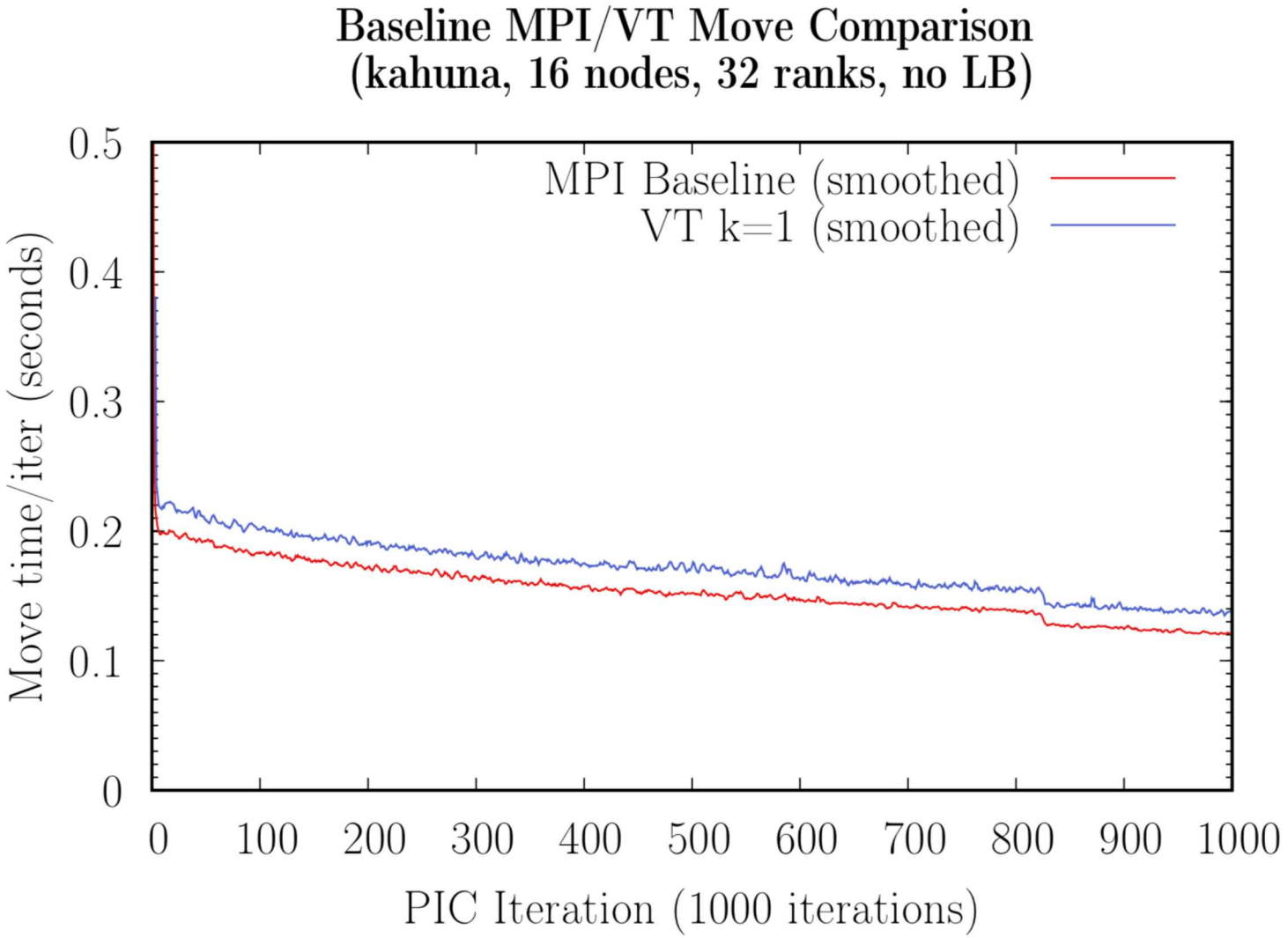
- MPI rank-based driver code (each rank drives physics for local block)
  - Example sequence of PIC operations:
    - particle_mover->setParticleDT()
    - particle_mover->injectParticles()
    - particle_mover->weightFields()
    - particle_mover->accelerateParticles()
    - Recursive particle move, send/recv, all-reduce, repeat
    - particle_mover->sortParticles()
    - particle_mover->updateCounts()
    - particle_mover->weightCharge()
  - In the MPI code, these are strictly sequenced, but some may be reordered, and possibly made concurrent

Sender-side Ordering with Epochs

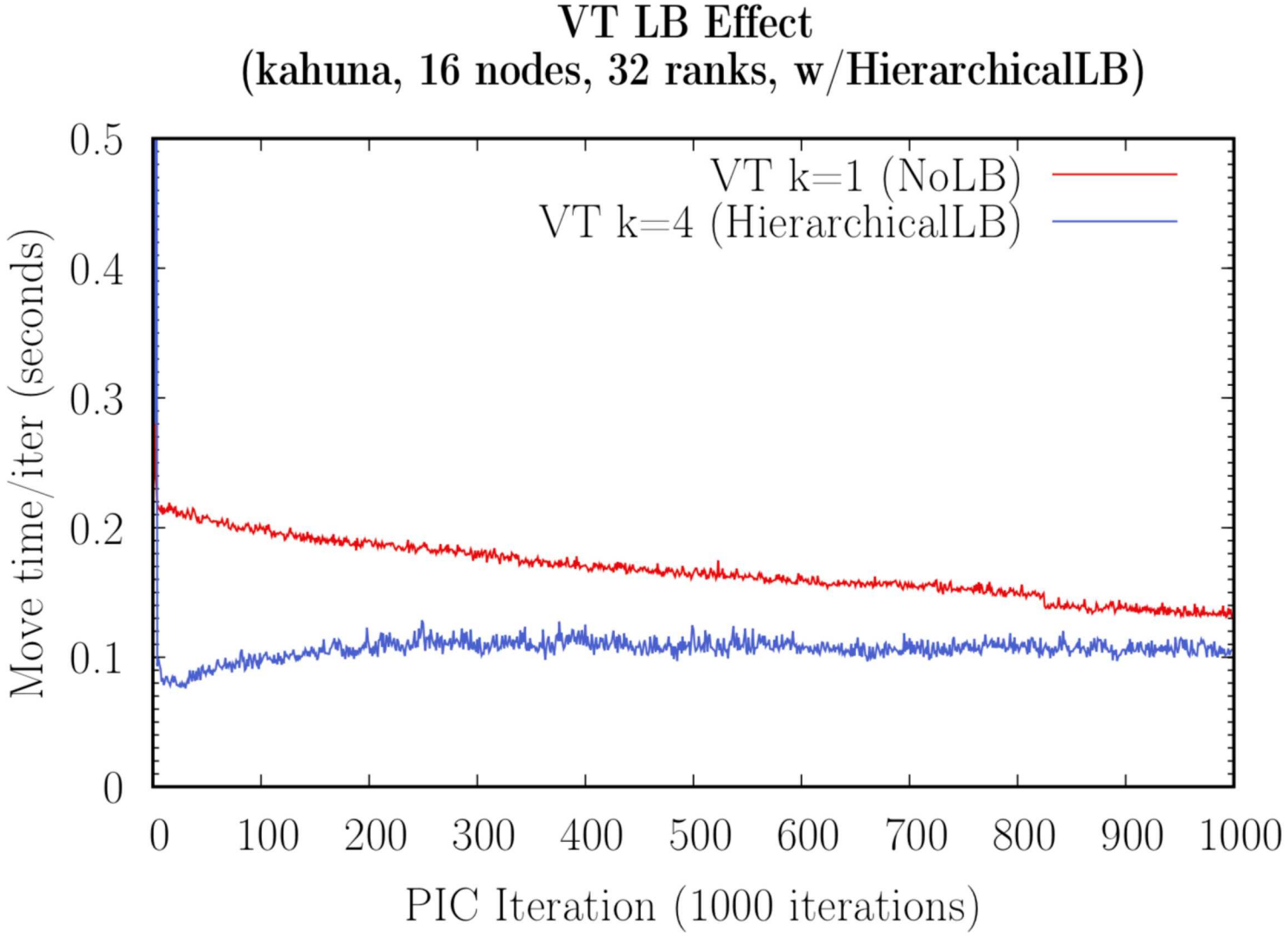
```
/* Every MPI rank */
auto col = vt::constructCollective<PIC_Col>(vt::Range(nranks,k));

/* for each PIC-move iteration */
vt::EpochType move_ep = vt::newCollectiveEpoch();
for (auto idx : localElms(col))
  col[idx].send<DoMoveMsg,moveFunc>(vt::makeMsg<DoMoveMsg>(move_ep));
vt::finishedEpoch(move_ep);

for (auto idx : localElms(col)) {
  vt::PendingSend sort = col[idx].send<SortMsg,sortFn>(sortMsg, move_ep);
  vt::PendingSend cnt  = col[idx].send<CountMsg,updateFn>(cntMsg, std::move(sort));
  col[idx].send<WeightMsg,weightCharge>(weightMsg, std::move(cnt));
  /* ... */
}
```

- If `vt::PendingSend` is not captured as a return value, the C++ destructor gets called immediately
  - Message gets sent with no extra cost
- VT creates a new rooted TD epoch to recursively track operations rooted at the message send
  - Dijkstra-Scholten engagement-tree termination detection for rooted epochs
  - 4-counter wave-based termination detection for collective epochs

Initial Results for Q1 Problem



Initial Results for Q1 Problem



Conclusion

Sandia National Laboratories

- DARMA project R&D is transitioning to a more application-driven approach for programmatic reasons
  - Building new distributed-memory programming model abstractions driven by application needs
- Research focused on runtime performance analysis and tuning, with a focus on distributed load balancing
  - EMPIRE-PIC needs a highly scalable distributed load balancing algorithm that is communication-aware
  - Philippe Pébaÿ will discuss our on-going research on distributed LB