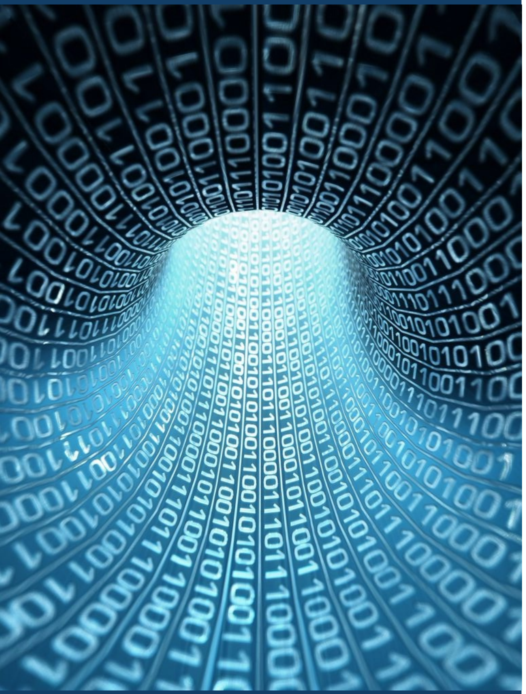
SAND2015-4043PE



Sandia National Laboratories

Exceptional service in the national interest

What is the DHARMA project?

Janine Bennett (PI), Jeremiah Wilke (Chief Architect)

Ken Franko, Hemanth Kolla, Paul Lin, Greg Sjaardema, Nicole Slattengren, Keita Teranishi

MAWG Meeting

2/5/2015

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

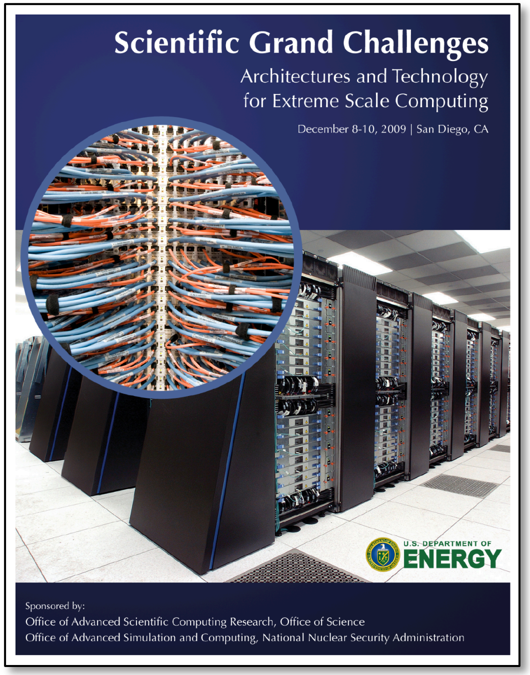
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

To start, the DHARMA project is not LOST ;)

Before the "What" a bit about the "Why"

We expect a lot from our programming models

- Programmability, expressiveness
  - Data parallelism: Same computation on different data
  - Task parallelism: Different computations on same or different data
- Performance, scalability
- Appropriate level of fault-tolerance
- Ability to debug/trace/analyze
- Portability
  - Abstractions separate code specification from optimization for different architectures
- Future-proof
  - How much of the application code needs to be rewritten when moving to new architectures?

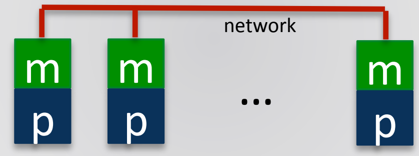


Performance and programmability are achieved by targeting an underlying abstract machine model

Machine model: PRAM/SMP

Programming model: threads

Machine model:
Bulk Synchronous Model



Programming model: MPI

Machine model: Hybrid Candidate Type Architecture (CTA)

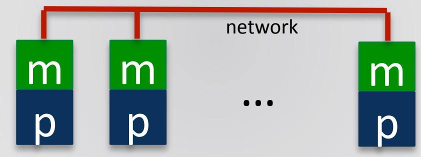
Programming model: Hybrid Bulk Synchronous MPI + X

Is hybrid bulk synchronous MPI+X future proof?

Machine model: PRAM/SMP

Programming model: threads

Machine model:
Bulk Synchronous Model



Programming model: MPI

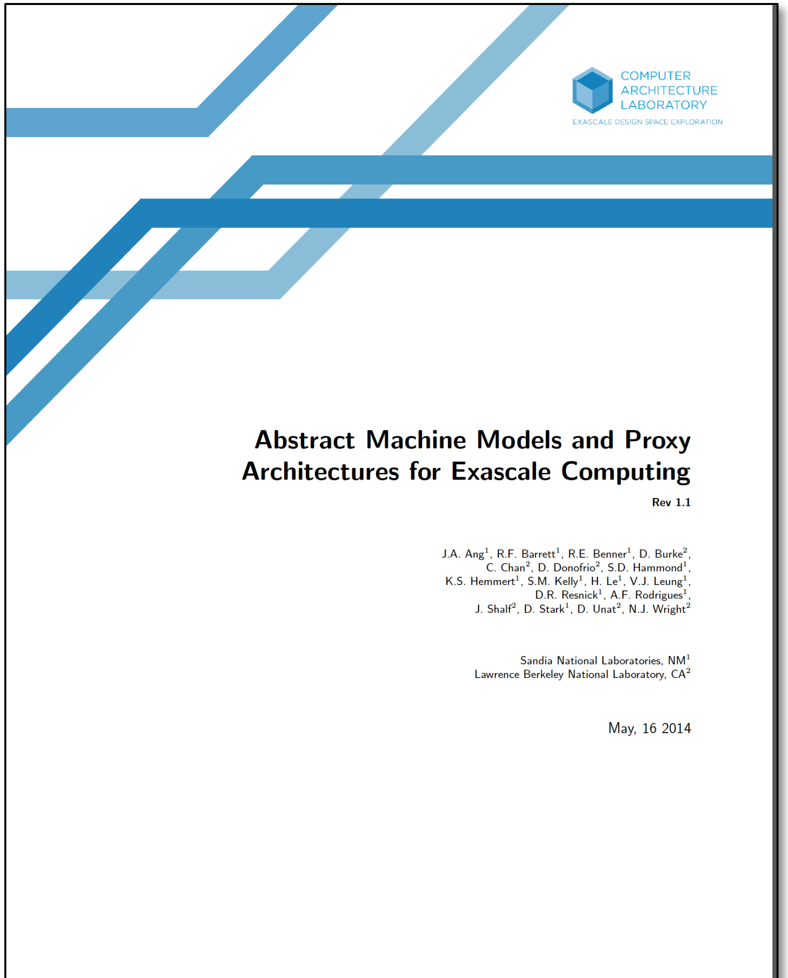
Machine model: Hybrid Candidate Type Architecture (CTA)

Programming model: Hybrid Bulk Synchronous MPI + X

- ✓ Programmability
- ✓ Appropriate level of fault tolerance
- ✓ Portability
- ✓ Performance
- ✓ Ability to debug/trace/analyze
- ? Future Proof

Consider the abstract machine model of an exascale node

Sandia National Laboratories


COMPUTER ARCHITECTURE LABORATORY
EXASCALE DESIGN SPACE EXPLORATION

Abstract Machine Models and Proxy Architectures for Exascale Computing

Rev 1.1

J.A. Ang[1], R.F. Barrett[1], R.E. Benner[1], D. Burke[2], C. Chan[2], D. Donofrio[2], S.D. Hammond[1], K.S. Hemmert[1], S.M. Kelly[1], H. Le[1], V.J. Leung[1], D.R. Resnick[1], A.F. Rodrigues[1], J. Shalf[2], D. Stark[1], D. Unat[2], N.J. Wright[2]

Sandia National Laboratories, NM[1]
Lawrence Berkeley National Laboratory, CA[2]

May, 16 2014

Overarching abstract machine model of an exascale node

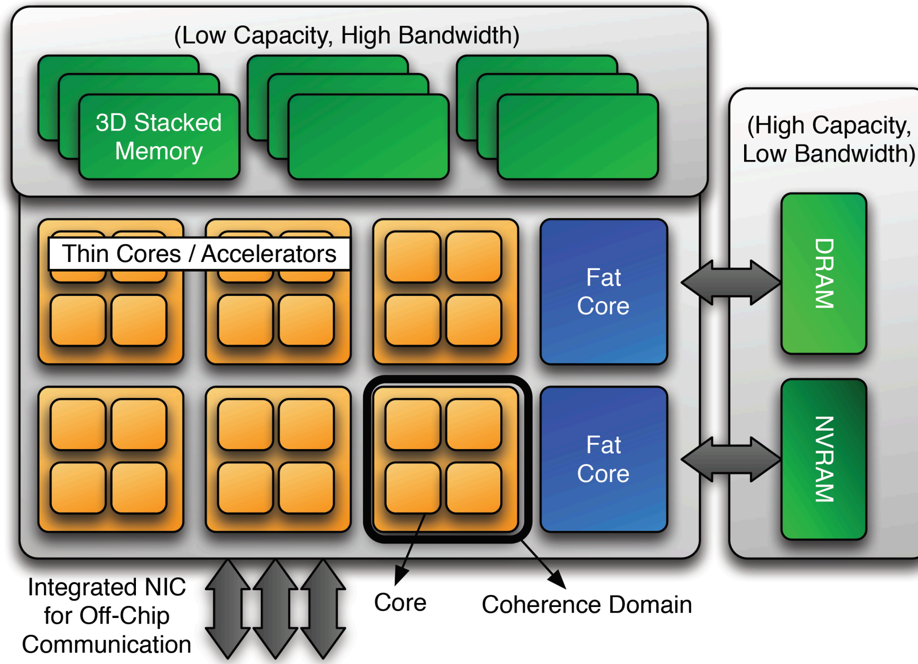


Image courtesy of www.cal-design.org

COMPUTER ARCHITECTURE LABORATORY
EXASCALE DESIGN SPACE EXPLORATION

This new abstract machine model introduces significant complexities

Challenges
▪ Increases in concurrency
▪ Deep memory hierarchies
▪ Increased fail-stop errors
▪ Performance heterogeneity ▪ Accelerators ▪ Thermal throttling ▪ General system noise ▪ Responses to transient failures

Overarching abstract machine model of an exascale node

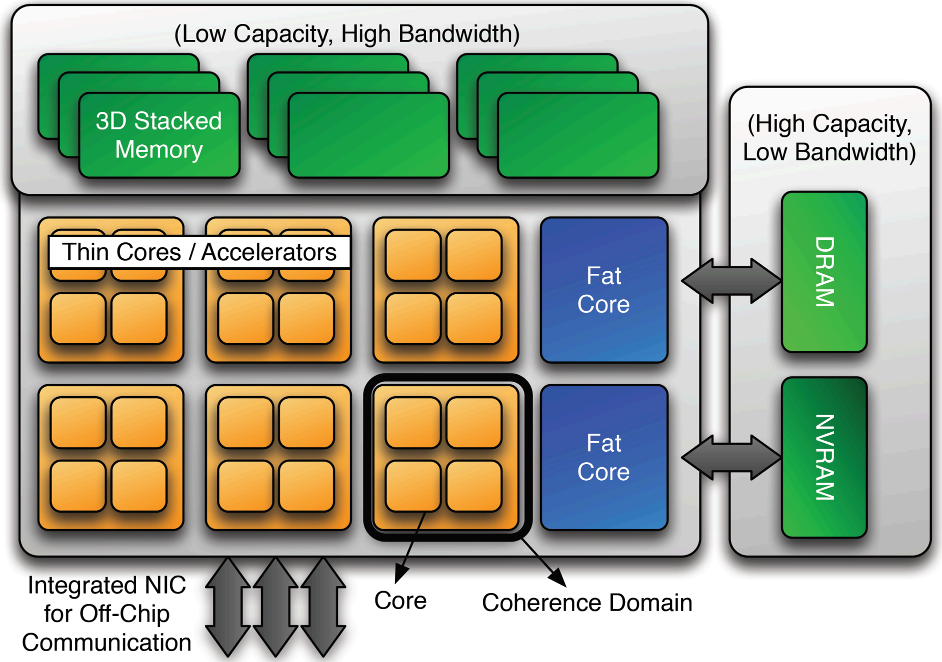


Image courtesy of www.cal-design.org

Bulk synchronous MPI+X does not address all the challenges posed by the exascale machine model

Sandia National Laboratories

Challenges
Increases in concurrency
Deep memory hierarchies
Increased fail-stop errors
Performance heterogeneity ▪ Accelerators ▪ Thermal throttling ▪ General system noise ▪ Responses to transient failures

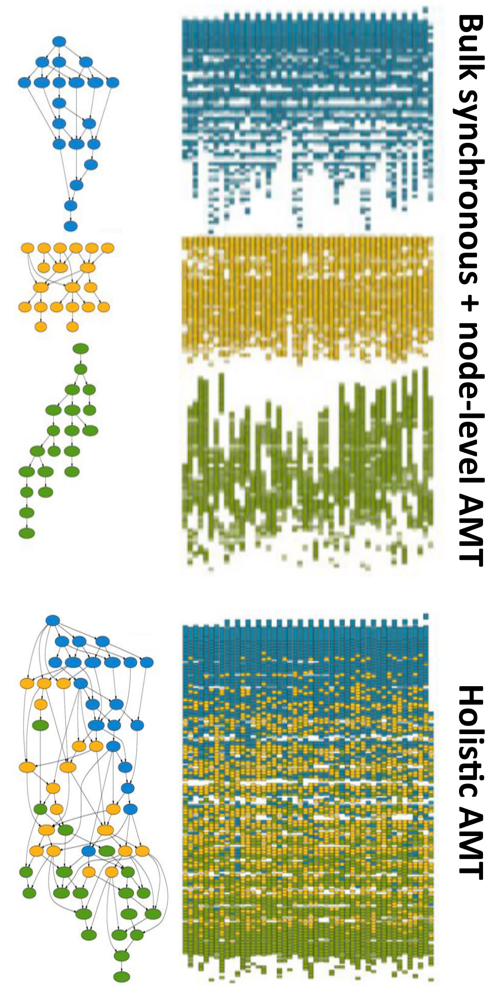
- Complexity of application code increases with proposed solutions

- Over-decomposition on node can help but does not solve the problem
- Algorithmic research required

Asynchronous many-task (AMT) programming models show promise against exascale challenges

Sandia National Laboratories

- Runtime systems show promise at sustaining performance despite node-degradation and failure
- Data flow programming model
  - Tasks are nodes in graph
  - Data dependencies are edges in graph
- Facilitate expression of task- and data-parallelism
- Active area of research
  - Charm++, DAGuE, DHARMA, HPX, Legion, OCR, STAPL, Uintah, …



Images courtesy of Jack Dongarra

DHARMA project: Distributed asyncHronous Adaptive Resilient Management of Applications

- **Project Mission:** Assess & address fundamental challenges imposed by the need for performant, portable, scalable, fault-tolerant programming models at extreme-scale

FY15 GOALS

Assess rich feature sets/usability/performance of existing AMT runtimes in the context of ASC workloads

Research in programmability, dynamic load-balancing, and fault-tolerance of AMT runtimes

DHARMA is a fundamental Hindu concept referring to
- the order and custom which make life and a universe possible
- the behaviors appropriate to the maintenance of that order

The classical Sanskrit noun DHARMA derives from dhr
- meaning to hold, maintain, keep

DHARMA project: Distributed asyncHronous Adaptive Resilient Management of Applications

- **Project Mission:** Assess & address fundamental challenges imposed by the need for performant, portable, scalable, fault-tolerant programming models at extreme-scale

Level 2	Assess rich feature sets/usability/performance of existing AMT runtimes in the context of ASC workloads
Level 2 & DHARMA runtime	Research in programmability, dynamic load-balancing, and fault-tolerance of AMT runtimes

DHARMA is a fundamental Hindu concept referring to

- the order and custom which make life and a universe possible
- the behaviors appropriate to the maintenance of that order

The classical Sanskrit noun DHARMA derives from dhr

- meaning to hold, maintain, keep

ASC ATDM Level 2 milestone description

- Overarching goal: Provide guidance to the code development road map for ATDM in the context of AMT, based on in-depth exploration using realistic proxies of ASC codes

- Key deliverables:
  - Implementations of one or mini-apps in three or more AMT runtimes
  - Analysis of the performance, programmability, and mutability of the AMT runtimes
  - An analysis of the interoperability of the runtimes
  - A report to inform the code development road map guiding the Sandia ASC code strategy for next generation platforms in the context of alternative programming models

Level 2 technical roadmap: programmability

- Does this programming model and RTS support the natural expression and execution of the ASC applications of interest
  - Implement miniapps in different RTS
    - To start miniAero in Charm++, Legion, Uintah
  - Qualitative questions for application developers
    - Rate abstractions, APIs
  - Quantitative data
    - Size of code
    - Length of time to code/optimize
    - Amount of code reuse from bulk-synchronous baseline implementation

Level 2 technical roadmap: programmability

- Does this programming model and RTS support the natural expression and execution of the ASC applications of interest
  - Implement miniapps in different RTS
    - To start miniAero in Charm++, Legion, Uintah
  - Qualitative questions for application developers
    - Rate abstractions, APIs
  - Quantitative data
    - Size of code
    - Length of time to code/optimize
    - Amount of code reuse from bulk-synchronous baseline implementation
- Where we would like your help:
  - Exploring more applications!
  - Characterizing the ASC workload

Level 2 technical roadmap: performance

- How long did it take to optimize the mini app code for performance and what were the performance gains?
- What are the scaling properties of the mini app in this RTS before and after performance optimization?
- How do the scaling properties and the runtime of the mini-app compare with the bulk-synchronous implementation?
- What are the scaling properties of the RTS itself?
- How performance portable is the RTS for ATSx-scale platform architectures? In other words, how shielded are the physics developers from changes in system architectures?
- How does the scaling of the mini app in this RTS change with task granularity and different levels of over-decomposition?
- How does this RTS provide support for dynamic load balancing?
- Can the application scientist directly control load balancing and/or provide load-balancing hints (e.g., physics/domain specific knowledge) to the RTS?
- How well does the RTS support fault containment and recovery?
- How does this RTS facilitate code coupling (e.g. in situ analysis and visualization, multi-physics?

Level 2 technical roadmap: performance

- Planned experiments:
  - Scaling studies
  - Work-granularity studies
    - Data: over-decomposition levels
    - Task: granularity (how much code is in the task)
  - Load balancing studies
    - System-induced imbalance
    - Application-induced imbalance

Level 2 technical roadmap: performance

- Planned experiments:
  - Scaling studies
  - Work-granularity studies
    - Data: over-decomposition levels
    - Task: granularity (how much code is in the task)
  - Load balancing studies
    - System-induced imbalance
    - Application-induced imbalance

- Where we would like your help:
  - Experimenting with additional applications implemented in the RTS
  - How does the RTS perform from a power and/or energy perspective?
  - What is the RTS impact on network behavior/saturation?

L2 technical roadmap: mutability

- How easy would it be to adopt this code base and make the changes necessary to suit ASC needs?
  - Identify key design decisions & associated impacts
  - Assess interoperability with other models/languages
  - Assess reusability/modularity of RTS components
  - Assess what a partnership strategy might look like
  - Describe state of tool chain (compiler, debugger, performance analysis)

L2 technical roadmap: mutability

- How easy would it be to adopt this code base and make the changes necessary to suit ASC needs?
  - Identify key design decisions & associated impacts
  - Assess interoperability with other models/languages
  - Assess reusability/modularity of RTS components
  - Assess what a partnership strategy might look like
  - Describe state of tool chain (compiler, debugger, performance analysis)

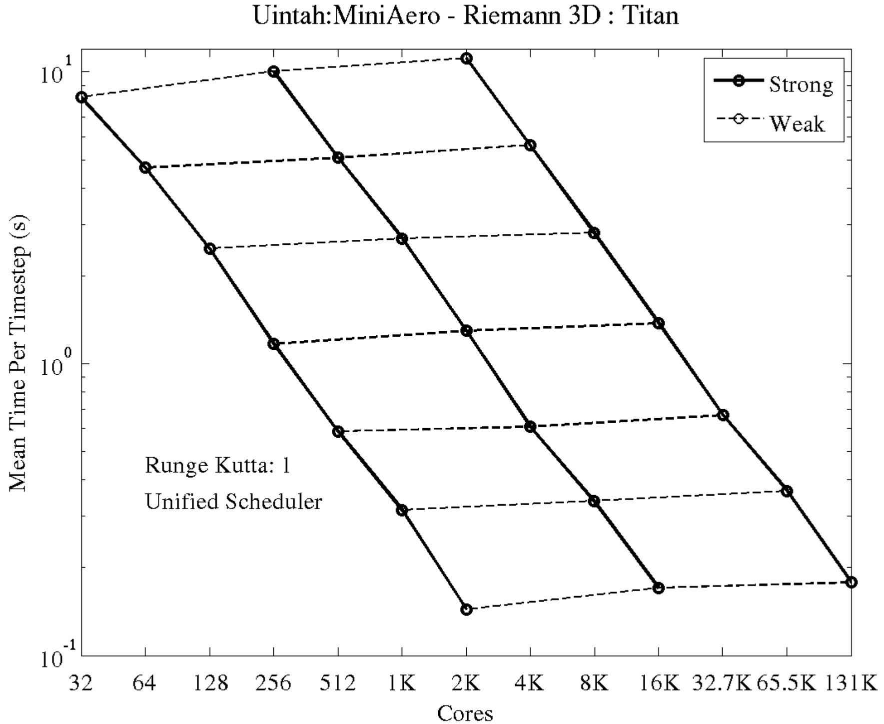
- Where we would like your help:
  - Again, answer these questions from the perspective of additional applications!
  - Identify integration path forward for RTS + node-level libraries (Kokkos, Qthreads, …)

L2 milestone implementation plan

- We considered many runtimes over the summer of FY14
  - Charm++, Legion, Uintah, STAPL, HPX, OCR, Swift/T
- We settled on Charm++, Legion, Uintah as our top three for the L2
  - Demonstrated science applications at scale
  - Maturity of runtime
  - Three very different implementations, APIs, sets of abstractions
  - Accessibility of team
- Coding Bootcamps
  - November 10-12 @ U. Utah (Uintah)
  - Dec 4-5 @ Stanford (Legion)
  - March 9-12 @ SNL CA (Charm++)
- Aim to be done with initial implementations by end of April
- Optimization/performance analysis/experiments April-July

L2 milestone status

- Uintah
  - Initial implementation of miniAero nearly complete
- Legion
  - Mesh generation making progress
- Charm++
  - Initial lecture online
  - Bootcamp in March
  - Start coding miniAero at bootcamp



Uintah initial scaling results

DHARMA project: Distributed asyncHronous Adaptive Resilient Management of Applications

- **Project Mission:** Assess & address fundamental challenges imposed by the need for performant, portable, scalable, fault-tolerant programming models at extreme-scale

Level 2	Assess rich feature sets/usability/performance of existing AMT runtimes in the context of ASC workloads
Level 2 & DHARMA runtime	Research in programmability, dynamic load-balancing, and fault-tolerance of AMT runtimes

DHARMA is a fundamental Hindu concept referring to

- the order and custom which make life and a universe possible
- the behaviors appropriate to the maintenance of that order

The classical Sanskrit noun DHARMA derives from dhr

- meaning to hold, maintain, keep

What makes support of fault tolerance in an AMT runtime challenging?

Bulk-synchronous approach is socialism

Task over-decomposition is anarchy

Bulk-synchronous	AMT
Everybody gets a fair share of work	Everybody takes as much work as they can do
Data dependencies appear in regular, well-defined locations	Data dependencies can appear anywhere
Collectives/synchronization signal WHEN dependencies are available	Data dependencies can appear anytime
When my work is done, my work is done	Termination detection is a challenging problem
Everyone agrees at beginning/end of iteration on global state	Everyone constantly agreeing on global state

In DHARMA a coarse-grained DAG defines stages of agreement for collections of tasks

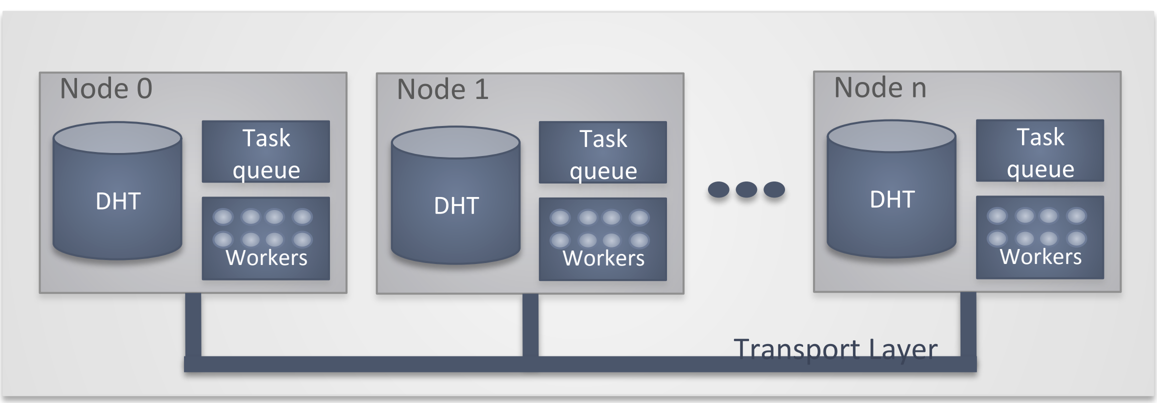
We have a distributed, resilient database consistency problem

- Group independent tasks into collections
- Agree at beginning of collection that all tasks are created, scheduled
- Agree at end of collection that there are no tasks left to run
- Agree at end of collection that all tasks expected were actually run
- Task collections can overlap

We do NOT force rigorous agreement on each database transaction

The DHARMA runtime comprises fault-tolerant components

- Distributed Hash Table (DHT): Manage where/when data exists
- Collection/Task Queue: Manage where/when tasks run
- Resilient Transport Layer: Manage termination detection and failed node detection
  - Fault-aware collectives: can detect errors and abort cleanly
  - Fault-tolerant collectives: always return valid result and rigorously agree on failed nodes



All runtime components are listening to system heartbeat implemented via fault-tolerant collectives

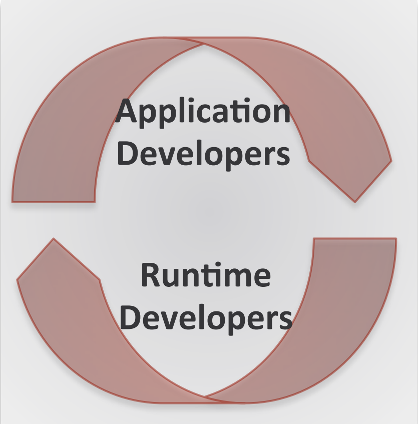
Core programmability questions

- What APIs and abstractions are needed to express the ASC workloads of interest?
- What constraints on data structures are good/bad?
- Do ASC developers feel their workloads are better expressed via:
  - Explicit task-graph vs. Implicit task-graph specification
  - Imperative vs. Declarative programming paradigms
  - User-specified vs. Automatic extraction of task-parallelism

Core programmability questions

- What APIs and abstractions are needed to express the ASC workloads of interest?
- What constraints on data structures are good/bad?
- Do ASC developers feel their workloads are better expressed via:
  - Explicit task-graph vs. Implicit task-graph specification
  - Imperative vs. Declarative programming paradigms
  - User-specified vs. Automatic extraction of task-parallelism

FY15 plans:



- L2 comparison study
  - Charm++, Legion, Uintah
- DHARMA v1.0 runtime philosophy
  - Use your own data structures, explicit task-graph, declarative, automatic extraction of task-parallelism

Core distributed load-balancing questions

- What is the right granularity of work?
  - What is the right level of over-decomposition?
  - How much work should a task comprise?
  - How do these numbers differ for load-balancing intra- and inter-node?
  - How do these numbers change for different applications & architectures?
- Which automatic load-balancing strategies work best for ASC applications?
- What are good mechanisms for allowing application developers
  - To directly control load-balancing?
  - To provide physics-based hints for load-balancing?
- What is the integration path forward for node-level, fine-grained parallelism libraries and distributed AMT runtimes?

Core distributed load-balancing questions

- What is the right granularity of work?
  - What is the right level of over-decomposition?
  - How much work should a task comprise?
  - How do these numbers differ for load-balancing intra- and inter-node?
  - How do these numbers change for different applications & architectures?
- Which automatic load-balancing strategies work best for ASC applications?
- What are good mechanisms for allowing application developers
  - To directly control load-balancing?
  - To provide physics-based hints for load-balancing?
- What is the integration path forward for node-level, fine-grained parallelism libraries and distributed AMT runtimes?

FY15 Plans:

- L2 milestone
  - Load-balancing performance analysis studies
  - Work granularity studies

Core fault tolerance questions

- How do you make an AMT flexible to different check-pointing/recovery strategies?
- What is required to transparently handle fail-stop node-crashes?
- What support mechanisms are needed for silent data corruption detection/correction?

Core fault tolerance questions

Sandia National Laboratories

- How do you make an AMT flexible to different check-pointing/recovery strategies?
- What is required to transparently handle fail-stop node-crashes?
- What support mechanisms are needed for silent data corruption detection/correction?

FY15 Plans:

- Build-out of DHARMA v1.0 runtime
  - Transparently handles fail-stop node crashes
  - Previous implementation in Structural Simulation Toolkit (ASC/CSSE FY14)
- L2 milestone

SST

http://sst.sandia.gov

Opportunities for collaboration

- Experimenting with additional applications
- Characterizing the ASC application workload
- Performance analysis and tools
- Exploring integration path forward with other areas of the software stack
  - Kokkos, data warehouse/Kelpie, Qthreads…
- Solvers, UQ