AIMES: Abstractions and Integrated Middleware for Extreme-Scale Science

Shantenu Jha, Matteo Turilli – Rutgers U

Jon Weissman, Francis Liu – U Minnesota

Daniel S. Katz, Zhao Zhang, Michael Wilde – U Chicago

MAGIC Meeting 3/4/2015

Funded by DOE ASCR under grants:
DE-FG02-12ER26115, DE-SC0008617, DE-SC0008651

Project Context

- Distributed Extreme-scale Science Applications
  - Access large distributed data sources
    - sensors and instruments, archives
  - Access diverse resource platforms
    - Leadership class, grids, clusters, clouds
  - Span big- and long-tail distributed science
  - Critical to the DOE mission
- Extreme scale, Extreme heterogeneity

Project Context (cont'd)

- Federating Distributed Resources is Essential to Extreme-Scale Computing. It allows us to:
  - Scale-out and scale-up
  - Exploit geo-locality
  - Hide complexity yet expose and exploit diversity
  - Support dynamism as a first-class entity

- But how to do it?

The Problem

- Lack of understanding, tools, and systems for how distributed applications can utilize federated resources
  - Need abstractions for:
    - applications, resources, middleware, infrastructure! (DCI), and how are they integrated?
  - Need models for:
    - applications, resources, middleware, infrastructure, and their composition

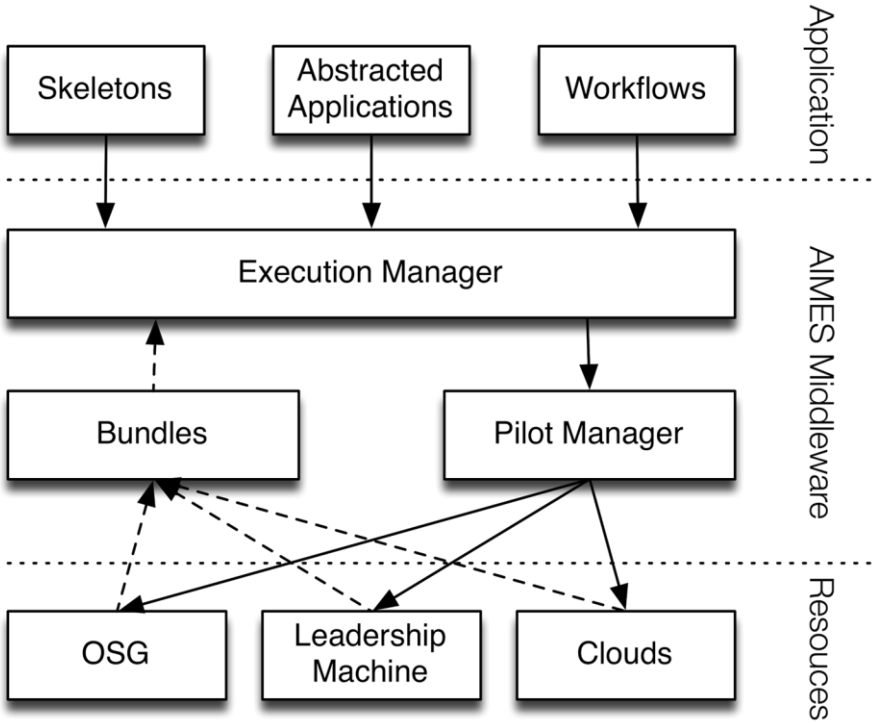
Solution: AIMES

- AIMES provides a laboratory to explore and reason:
  - abstractions -> models -> knowledge discovery
  - “how will my application perform on this DCI?”
  - “how can I best adapt my application to a DCI?”
  - “how can a DCI best adapt to my application?”
  - “why did the system allocate this DCI to my application?”

- AIMES laboratory can be embedded:
  - components of AIMES can be used by actual tools, systems, and applications

AIMES Abstractions

- Application skeletons
- Pilots/Execution Strategies
- Bundles



AIMES Objectives

- Enable reasoning about executing distributed workloads
  - model bundles, execution strategies, and skeletons
  - what “variables” matter most … matter least

- To do this: prototype AIMES software stack in a federated environment
  - deploy abstractions and middleware and assess
  - explore methods of integration across layers
  - collect qualitative/quantitative evidence towards models

Inside AIMES: Skeletons

- Goal: Hide application complexity while capturing essential characteristics
- **Application Skeleton** is a simple yet powerful tool to build synthetic applications that represent real applications, with similar performance.
- Design and Implementation:
  - Applications are represented by a compact set of parameters
  - Bag of Tasks, (iterative) map-reduce, and (iterative) multistage workflow applications
  - Multiple targets including: AIMES, Swift, Pegasus, Shell

Micro Result: Skeletons Are Accurate

- Montage, Blast, Cybershake

TABLE II. TIME-TO-SOLUTION COMPARISON OF SKELETON MONTAGE AND REAL MONTAGE (SECONDS)

	mProject	mImgtbl	mOverlaps	mDiffFit	mConcatFit	mBgModel	mBackground	mAdd	Total
Montage	282.3	139.7	10.2	426.7	60.1	288.0	107.9	788.8	2103.7
Skeleton	281.8	136.8	10.0	412.5	59.2	288.1	106.2	781.8	2076.4
Error	-0.2%	-2.1%	-0.2%	-3.3%	-1.5%	0.03%	-1.6%	-0.9%	-1.3%

TABLE IV. TIME-TO-SOLUTION COMPARISON OF SKELETON BLAST AND REAL BLAST (SECONDS)

	split	formatdb	blastp	merge	Total
BLAST	74.4	82.1	1996.3	35.9	2188.7
Skeleton	72.9	81.6	2028.9	36.3	2219.7
Error	-1.9%	-0.6%	1.6%	1.1%	1.4%

TABLE VI. TIME-TO-SOLUTION COMPARISON OF SKELETON CYBERSHAKE AND REAL CYBERSHAKE (SECONDS)

	Extract	Seis	PeakGM	Total
CyberShake	571.5	2386.5	81.5	3039.4
Skeleton	586.3	2443.3	83.3	3112.9
Error	2.6%	2.4%	2.3%	2.4%

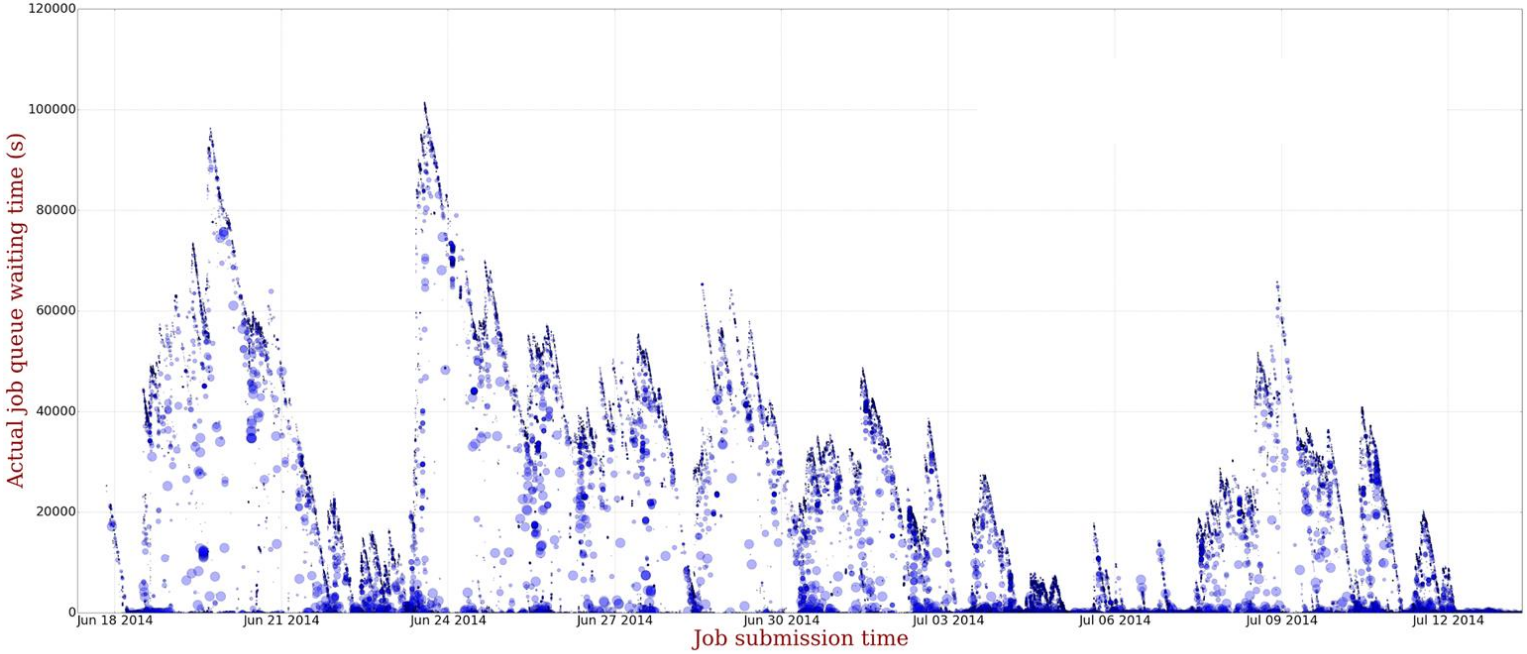
Inside AIMES: Bundles

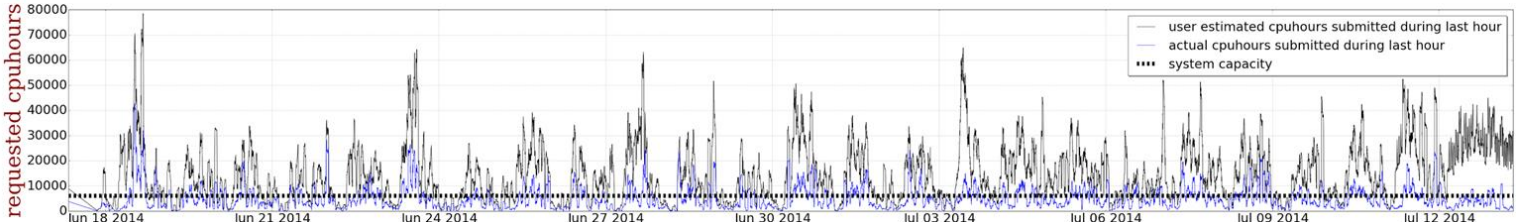
- Goal: to characterize heterogeneous resource aggregates
  - defines a unifying representation of het resources
  - hides platform-specific details
  - enables automatic, on-demand selection of resources by providing resource information

- Design and implementation
  - Compute bundles: NSF XSEDE and FutureGrid, DOE

Micro Result: Bundle Characterization

- XSEDE: large core count priority, waiting time skew (very short or very long)



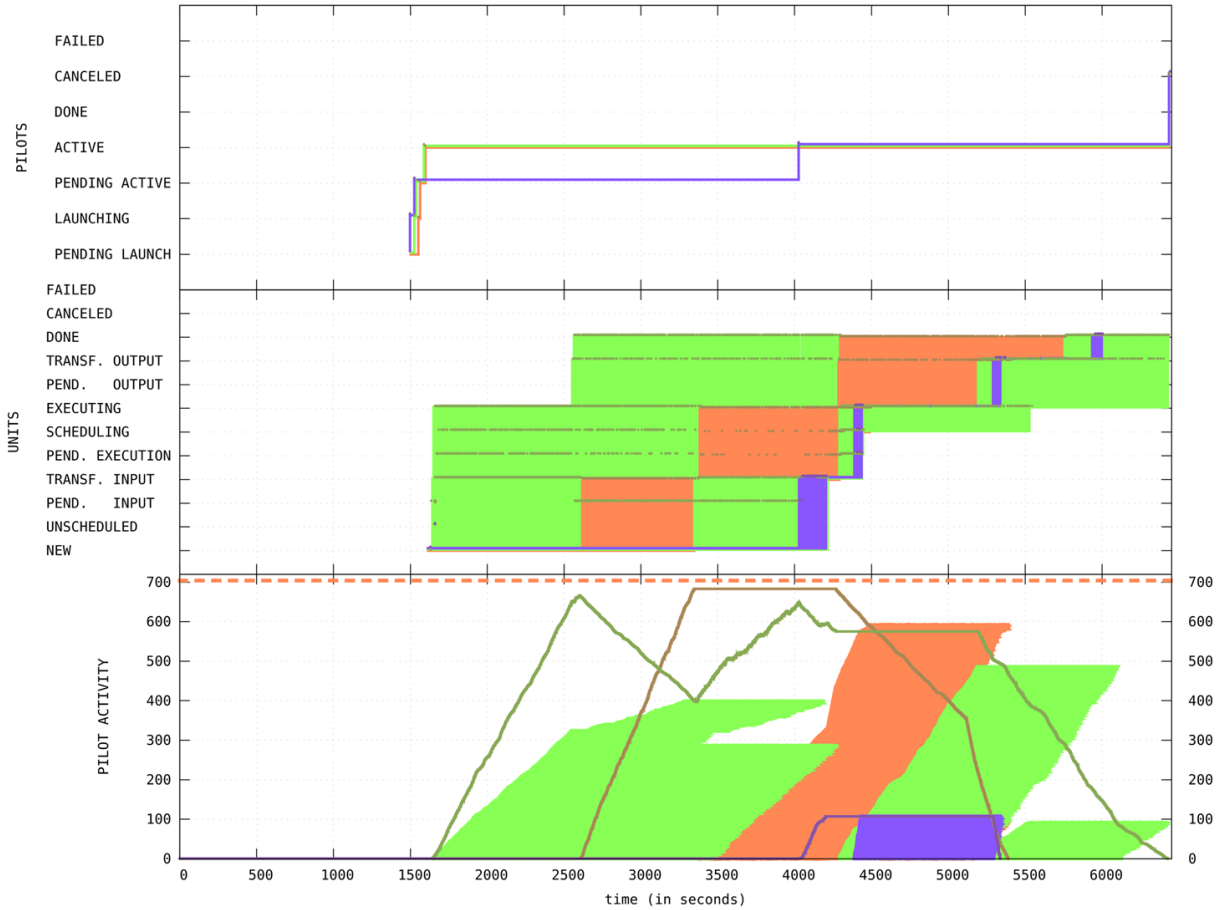


Inside AIMES: Pilots

- Goal: to characterize workload description and execution requirements on federated resources
  - qualitative: modeling the concept of ‘execution strategy’
  - quantitative: defining key choices when executing a given workload
- Design and Implementation:
  - pilot-based overlay of heterogeneous resource federation
  - consistent representation of execution strategies
  - transparent workload placement and scheduling algorithms across multiple pilots

End to End Result: Federated Execution

- **Overlay** based federation of three resources.
- **Late-binding**: CUs (tasks) are dynamically bound to a resources by means of a backfilling scheduling algorithm.



2048 CUs executed on 3 708-core pilots on Trestles, Stampede, and Gordon.

Using skeletons, bundles, pilots, execution manager.

AIMES Milestones

- Year 1: Functional integration of components: SC13
- Year 2: Steps towards quantification: SC14
- Year 3: Deeper quantification and reasoning: SC15
  - expansion of scale and heterogeneity
- Take-away thus far: engineering success has provided early evidence that we are on the right track

Year 3

- Scale:
  - Multi-site OSG
- Heterogeneity:
  - Expand to network and storage resources
- Applications:
  - Data-dependent workflows
- Deeper integration:
  - SWIFT

The Future

- From Modeling to Models:
  - Modeling of execution **must** include abstractions and models of DCI components and their federation
    - How to compose models?
    - Does planning and execution improve with DCI models?
- From Sensing to Actuation:
  - Resource discovery and actuation
  - How to architect infrastructure for specific performance and requirements?
    - Design principles and architectures for next generation of DCI

Thank You!

Questions?