Intelligent Optimization of Parallel and Distributed Applications

USC/ISI:
Jacqueline Chame
Chun Chen
Ewa Deelman
Yolanda Gil
Mary Hall
Kristina Lerman
Yoonju Lee Nelson

USC (Physics/CS):
Bhupesh Bansal
Aiichiro Nakano
Priya Vashishta

OSU:
Umit Catalyurek
Vijay Kumar
Tahsin Kurc
Joel Saltz
Ashish Sharma

This research has been partially funded by NSF NGS and CSR.

Motivation

Historical Perspective

- Highly tuned applications are too hard to develop, port
- Complexity leads to fragile applications & system software
- Ad hoc approaches, not formalized
- Community knowledge exists in the minds of too few people
- Fragmentation of community, duplication of effort

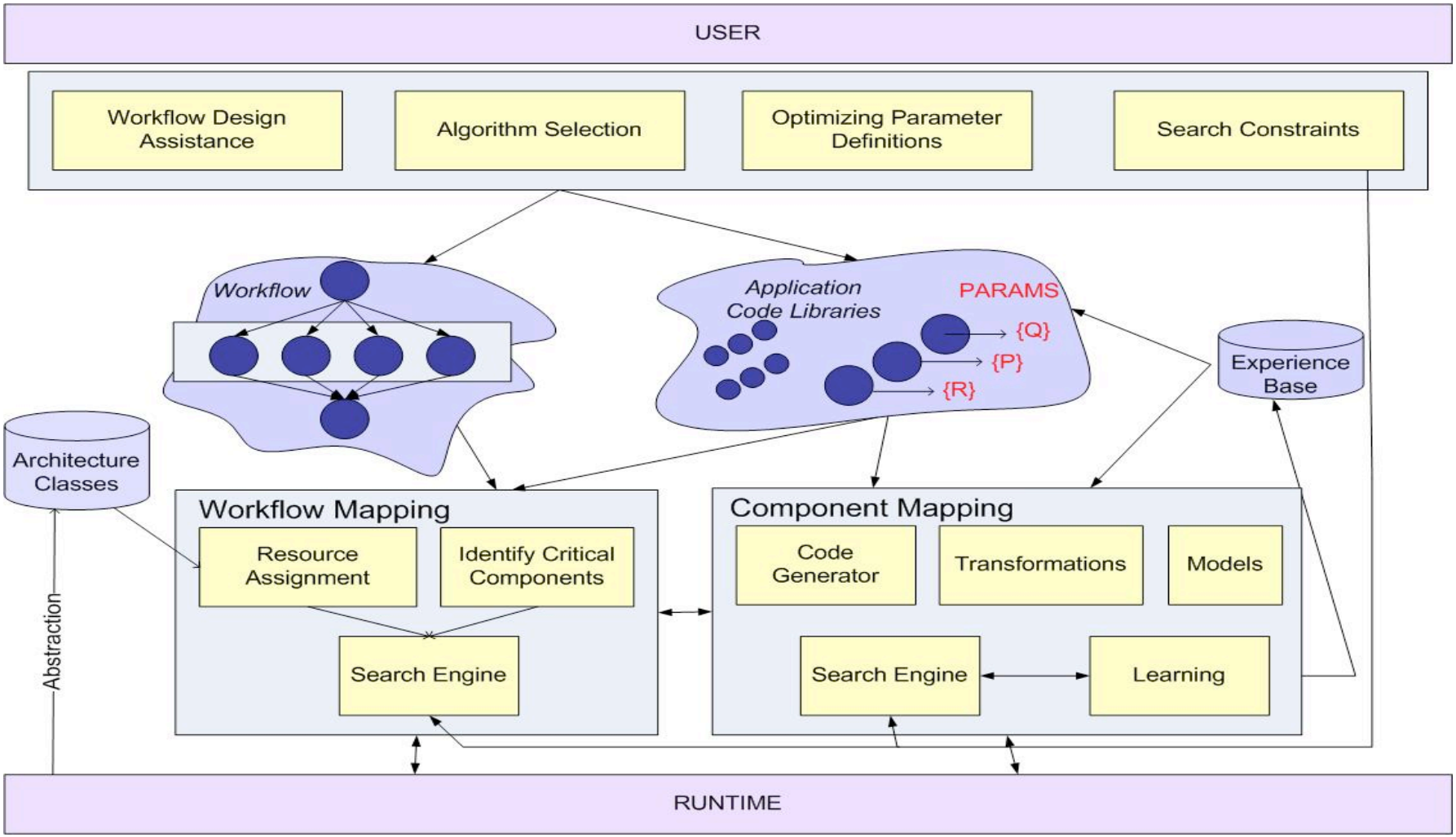
How to Move Forward

- Systematize the process of constructing and tuning applications from existing components or patterns
- Build tools that form a foundation to which many can contribute and improve
- Organize the community to work together

USC Viterbi
School of Engineering

THE OHIO STATE UNIVERSITY

System Design



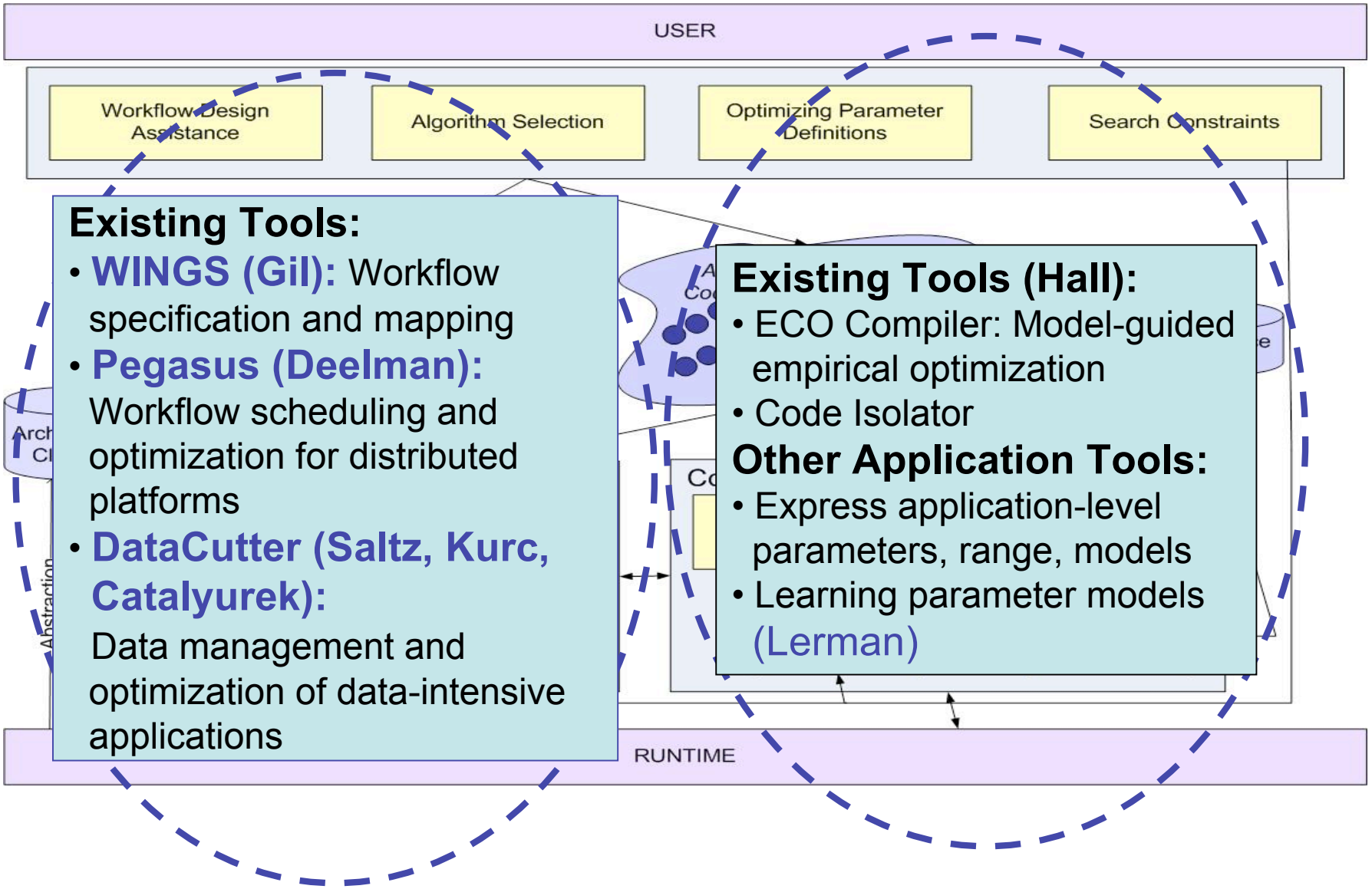
Key Concepts

- A systematic strategy for composing application components into workflows
- Search for the most appropriate implementation of both components and workflows
- Component optimization
  - Select among implementation ***variants*** of the same computation
  - Derive integer values of optimization ***parameters***
  - Only search promising variants and a restricted parameter space
- Workflow optimization
  - Knowledge-rich representation of workflow properties

USC Viterbi
School of Engineering

THE OHIO STATE UNIVERSITY

System Design



Goals and Methodology

User's perspective:

- Optimize the performance of workflow execution

Systems perspective:

- Enable runtime environment to select parameter values to optimize performance
- To learn a performance model for automation

How it works:

- **WINGS**: Semantic description for a workflow that ***exposes performance parameters*** to the runtime system.
- **Runtime system**: ***Hierarchical runtime*** environment:

  Pegasus/Condor: coarse-grain scheduling of workflow and data transfers

  DataCutter: fine-grained scheduling / high performance / task parallelism

USC Viterbi School of Engineering

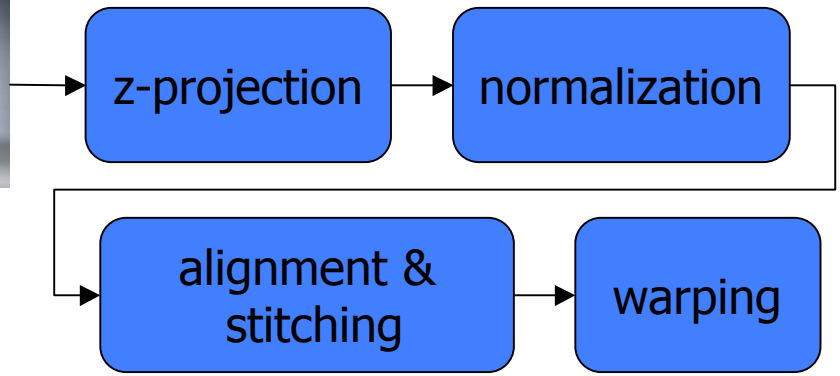
THE OHIO STATE UNIVERSITY

Focus on Parameter Selection for Biomedical Imaging Workflow

- Combined infrastructures
  - Pegasus + Wings were already combined
  - Learned how to run DataCutter jobs from Pegasus
- Selected a Biomedical Imaging Application for experimentation
  - Already runs under DataCutter
  - First, a WINGS representation (workflow template, instance
  - Then, execution under Pegasus
- Pairwise experiments
  - Component optimization of MD simulation (see Nelson talk at HIPS!), and now warping component

Designing and Parameterizing a Workflow for Optimization

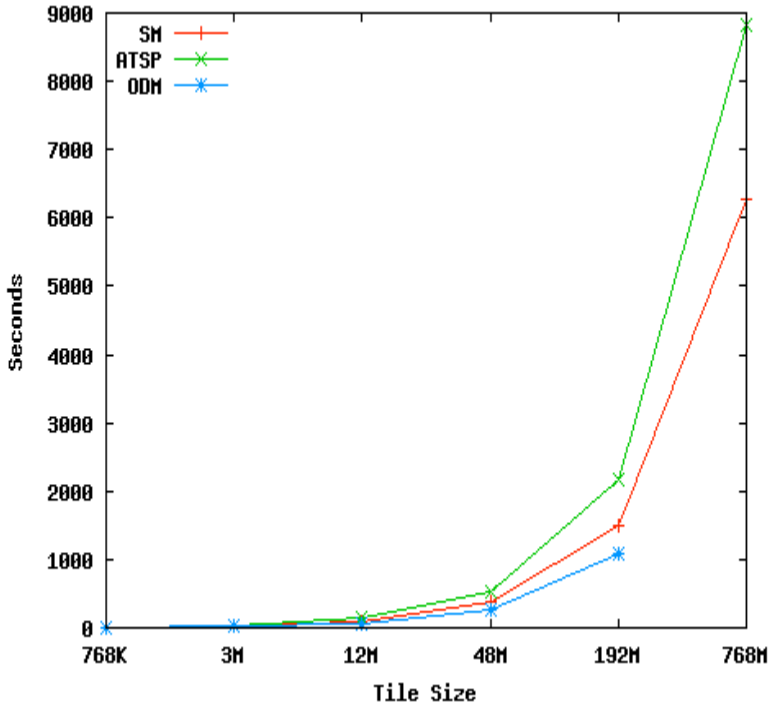
Application: Biomedical image analysis



Sample workflow for image correction

Each component is a DataCutter filter, enabling processing of out-of-core image data.

- ***Application-level parameters***: chunk size
- ***Middleware-level parameters***: number of DataCutter filter copies per node

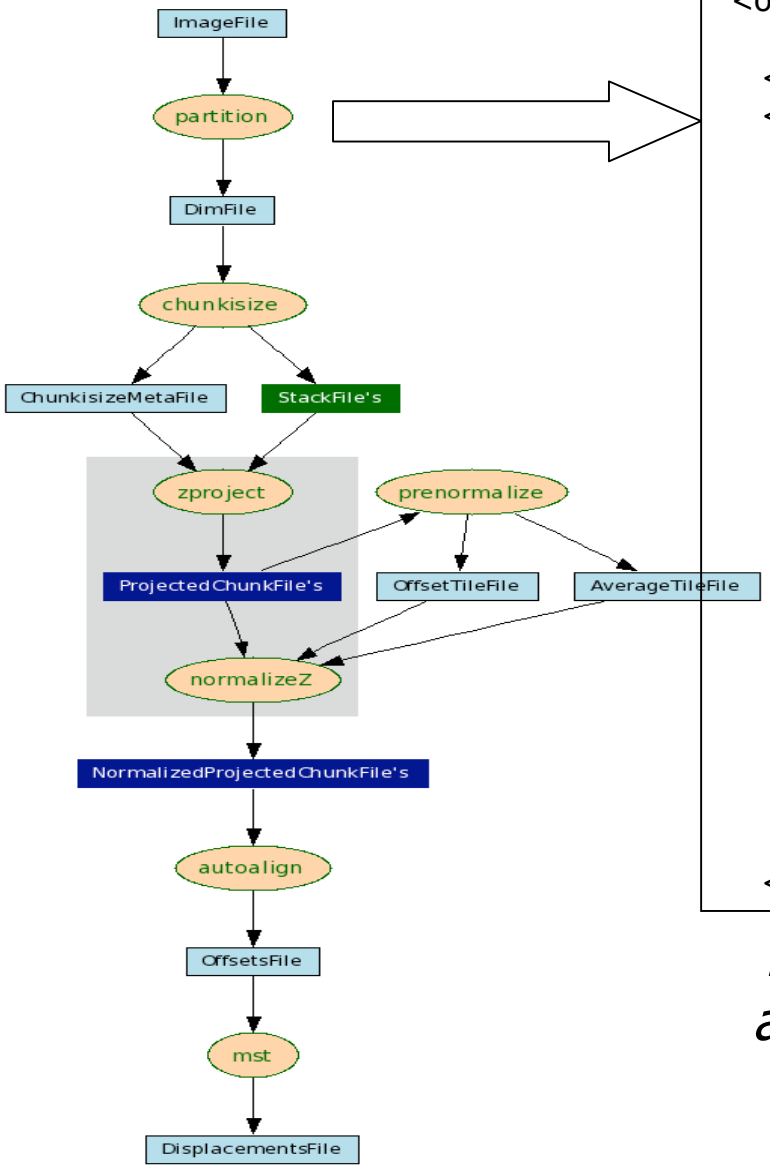


Influence of chunk size on performance of warping stage (for different warping algorithms)

USC Viterbi School of Engineering

The Ohio State University

Workflow Representation in WINGS

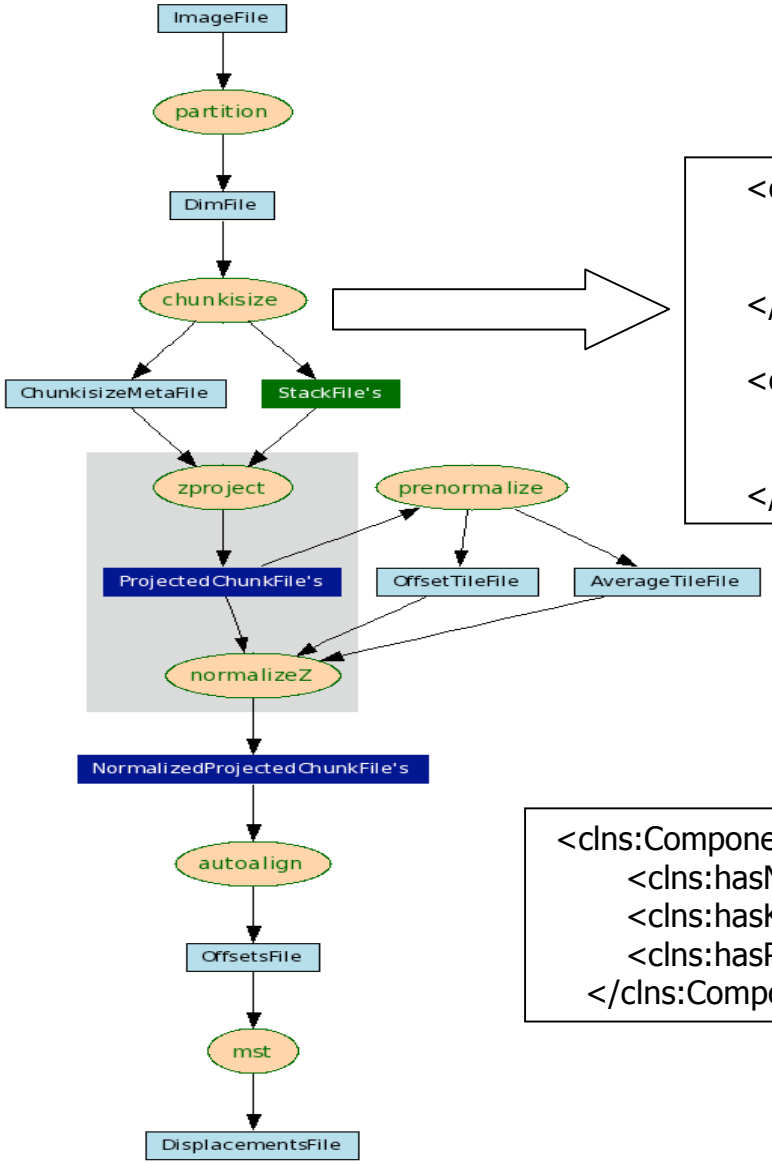



```
<owl:Class rdf:ID="partition">
    <rdfs:subClassOf rdf:resource="&clns;ComponentType"/>
  </owl:Class>
  <partition rdf:ID="partition1">
    <clns:hasVersion rdf:datatype="&xsd;string">1</clns:hasVersion>
    <clns:hasArguments>
      <rdf:List>
        <rdf:first rdf:resource="#partition_arg1"/>
        <rdf:rest rdf:resource="&rdf;nil"/>
      </rdf:List>
    </clns:hasArguments>
    <clns:hasInputs>
      <rdf:List>
        <rdf:first rdf:resource="#partition_Input_ImageFile"/>
        <rdf:rest rdf:resource="&rdf;nil"/>
      </rdf:List>
    </clns:hasInputs>
    <clns:hasOutputs>
      <rdf:List>
        <rdf:first rdf:resource="#partition_Output_DimFile"/>
        <rdf:rest rdf:resource="&rdf;nil"/>
      </rdf:List>
    </clns:hasOutputs>
  </partition>
```

Extend the WINGS base ontology to represent application-specific components and data types

THE OHIO STATE UNIVERSITY

Workflow Representation in WINGS




```
<cIns:ArgumentAndPrefix rdf:ID="chunkisize_arg5">
   <cIns:hasPrefix rdf:datatype="&xsd;string">-chunksizeX</cIns:hasPrefix>
   <cIns:hasArgument rdf:resource="#chunkisize_Output_ChunksizeX"/>
</cIns:ArgumentAndPrefix>

<cIns:ArgumentAndPrefix rdf:ID="chunkisize_arg6">
   <cIns:hasPrefix rdf:datatype="&xsd;string">-chunksizeY</cIns:hasPrefix>
   <cIns:hasArgument rdf:resource="#chunkisize_Output_ChunksizeY"/>
</cIns:ArgumentAndPrefix>
```

Exposing application-level parameters to the runtime system

```
<cIns:ComponentProfile rdf:ID="chunkisize_Machine_Count_Profile">
     <cIns:hasNamespace rdf:datatype="&xsd;string">condor</cIns:hasNamespace>
     <cIns:hasKey rdf:datatype="&xsd;string">machine-count</cIns:hasKey>
     <cIns:hasProfileVal rdf:resource="#chunkisize_Output_NOfChunks"/>
  </cIns:ComponentProfile>
```

Exposing middleware-level parameters to the runtime system

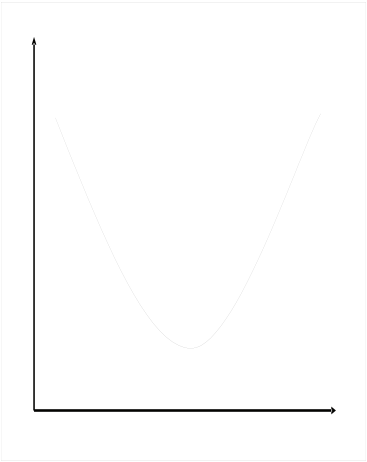
USC Viterbi School of Engineering

THE OHIO STATE UNIVERSITY

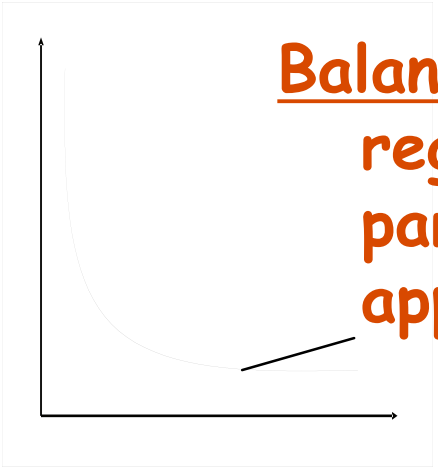
Switching Gears: Component Optimization

- Combine workflow optimization with component optimization
- Two approaches:
  - Autotuning compiler
  - User-guided autotuning (semi-autotuning?)
- General strategy
  - Model-guided empirical optimization

Examples of Models and Heuristics



Capacity:
tile size,
buffer size,
app-level parms

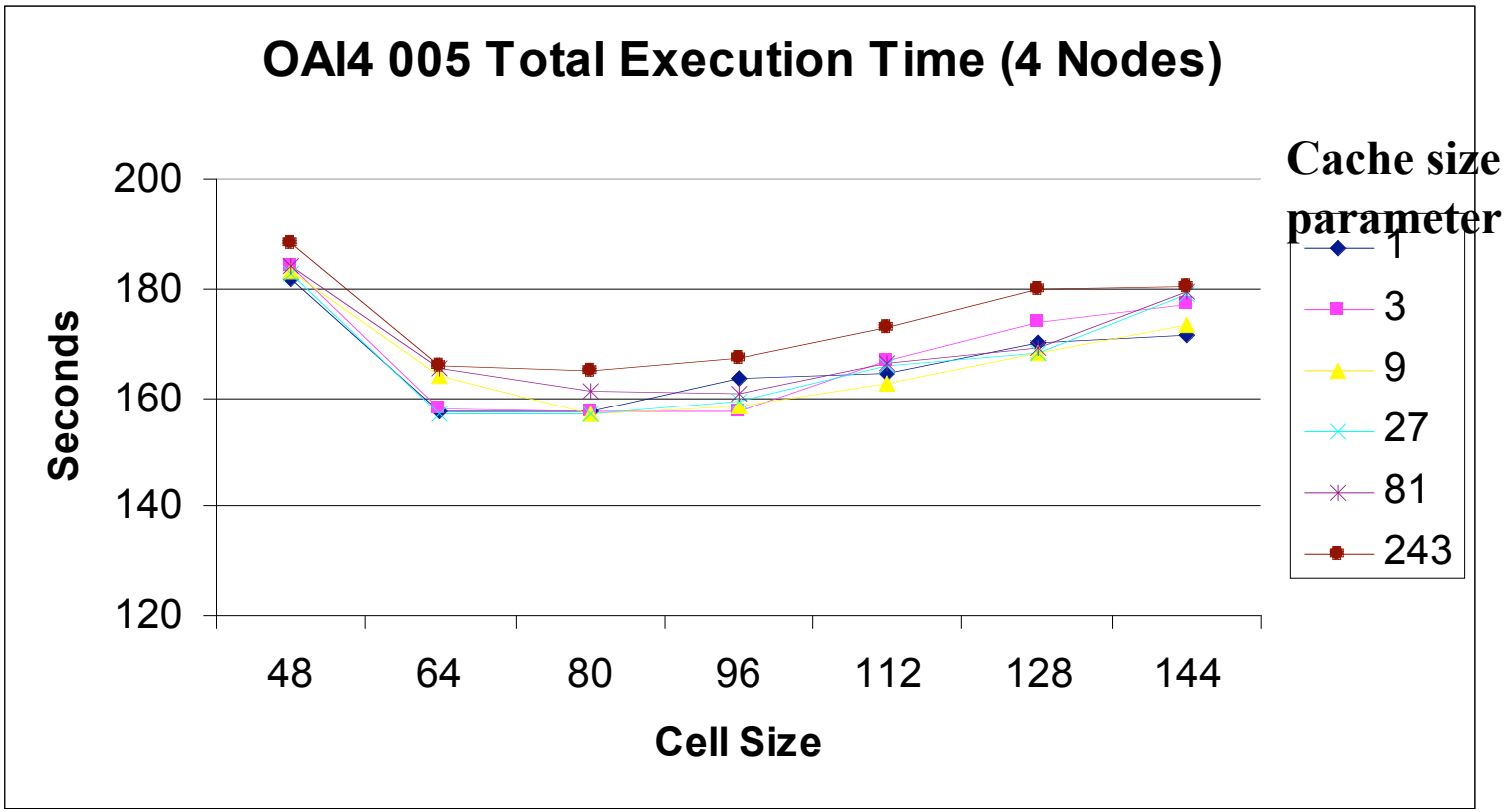


Balance:
register/FPU,
parallelization,
app-level parms

- Compiler models:
  - Reuse analysis to model application behavior under different loop orders, tiling strategies and unroll strategies
- Compiler heuristics:
  - Only use S-1/S sets for retained data in S-way set associative cache
- Functional models of parameters
  - Load balance as a function of computation partitioning

How well do models predict performance?

Execution on Linux cluster



Model-Guided Performance Tuning of Parameter Values: A Case Study with Molecular Dynamics Visualization, Y. Nelson, B. Bansal, M. Hall, A. Nakano, and K. Lerman, *Proceedings of the Workshop on High-Level Parallel Programming Models and Supportive Environments*, April 2008.

USC Viterbi School of Engineering

THE OHIO STATE UNIVERSITY

Concluding Remarks

- Three core technical ideas in project
  - **Compiler technology:** Modular compilers, systematic approach to optimization, empirical search, ***goal is hand-tuned performance***
  - **Components:** Tunable, automatically-generated XML-based interfaces, knowledge representations, more empirical search
  - **Systematic:** Based on machine learning, knowledge representation
- Focus on long-term evolutionary path
- ... And community organization