Exploring Tuning Strategies for Quantum Chemistry Applications

Lakshminarasimhan Seshagiri, Meng-Shiou Wu, Masha Sosonkina
Ames Laboratory , Ames, IA 50011
Zhao Zhang
Iowa State University, Ames, IA 50011

* This work was supported in part by the National Science Foundation Grants NSF/OCI-0749156 and NSF/CHE-0535640; and in part by Iowa State University of Science and Technology under the contract DE-AC02-07CH 11358 with the U.S. Department of Energy.

Outline

- Motivation
- Introduction to GAMESS and existing adaptation structure using NICAN
- Methodology
- Performance Results
- Tuning Strategy
- Conclusions and Future Work

Motivation

- Computational Chemistry application performance depends on
  - Input parameter combinations
  - Underlying hardware configuration
- Adaptation to varying system conditions is required for consistently good performance.
- Application performance analysis required to understand effect of input parameters and system configuration on application performance.
- Analysis helps to design a tuning strategy for such applications.

Introduction

Ab initio Quantum Chemistry Applications

- Studies properties of molecules (energy, geometry etc)
- Based on Schrödinger equation.
- Schrödinger equation can be solved (only) approximately
  - *semi empirical - uses experimental measurements*
  - *ab-initio - collection of mathematical methods*
- Other scientific applications based on *ab-initio* methods includes GAMESS, NWCHEM, MOLPRO

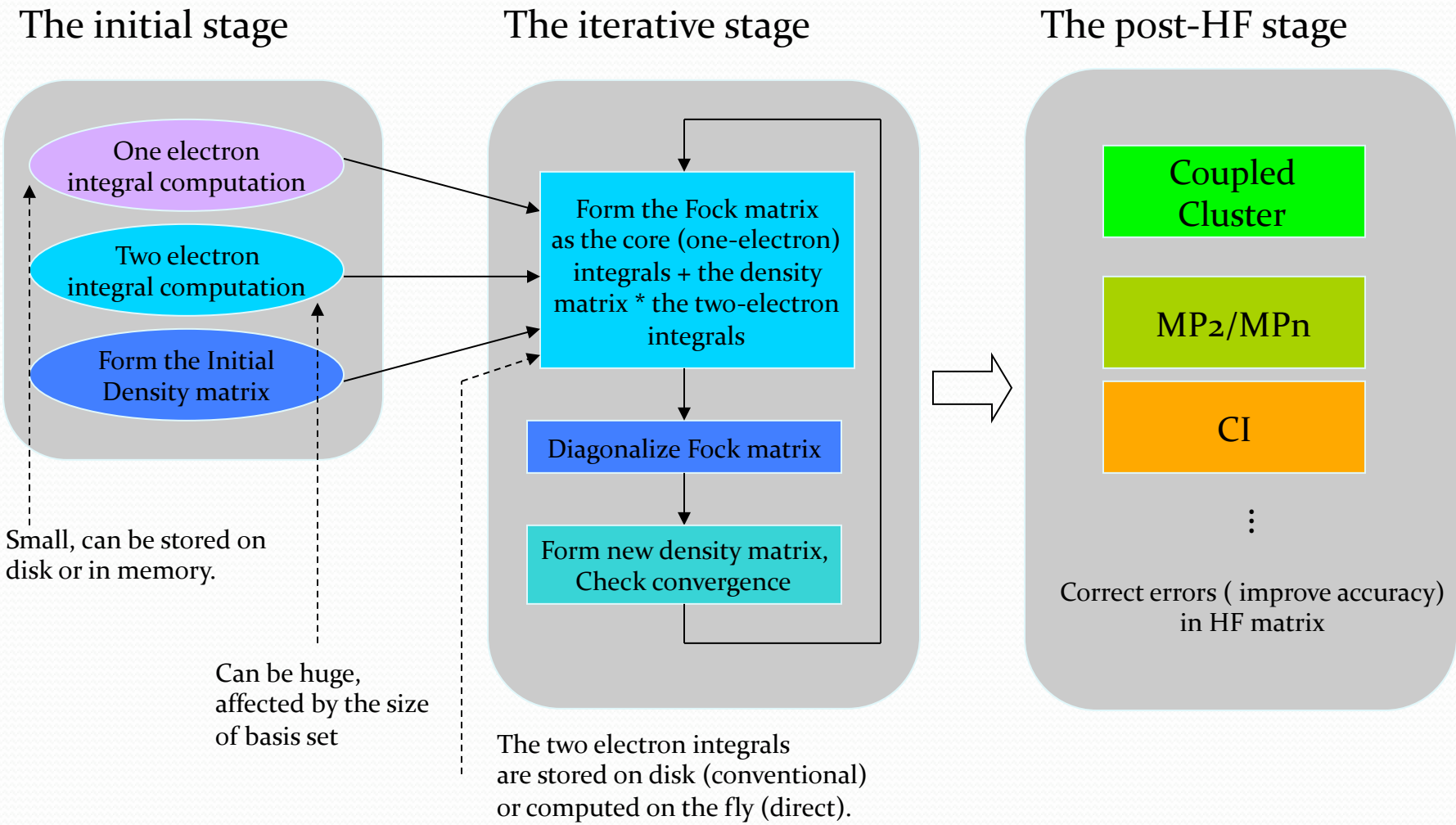
Introduction

GAMESS

- General Atomic and Molecular Electronic Structure System
- is generic *ab initio quantum chemistry calculation package*
- calculates wide range of Hartree-Fock (HF) wave functions (RHF, ROHF, and UHF)
- uses Self-Consistent-Field (SCF) method (with *direct and conventional implementations)*
  - *direct - recomputes integrals on-the-fly for each iteration (memory and* CPU intensive)
  - *conventional - computes integrals once, stores on disk, and reuses for* each iteration (I/O intensive)

Introduction

Computation Process



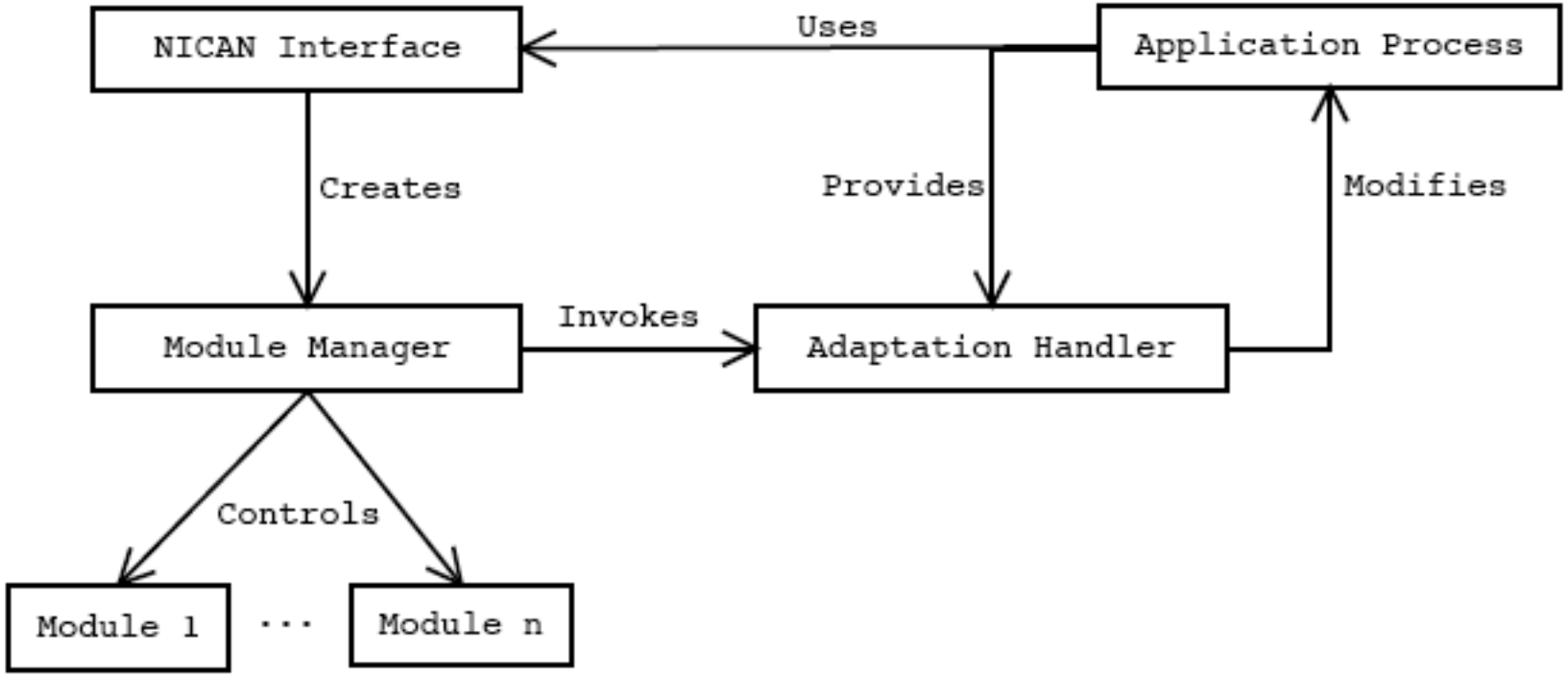
Introduction

- Two patterns of execution (*direct and conventional)* favor different computational resources
- Need for efficient execution of GAMESS jobs and analysis of system resources: memory, I/O, architecture (SMP)
- Incorporating self-scheduling into GAMESS or manual analysis by the user is infeasible
- Modern schedulers (PBS, LoadLeveler, LSF, etc..) incapable to “peek” into application’s execution
- **Integrate GAMESS with application level middleware (*NICAN)***

Introduction
NICAN

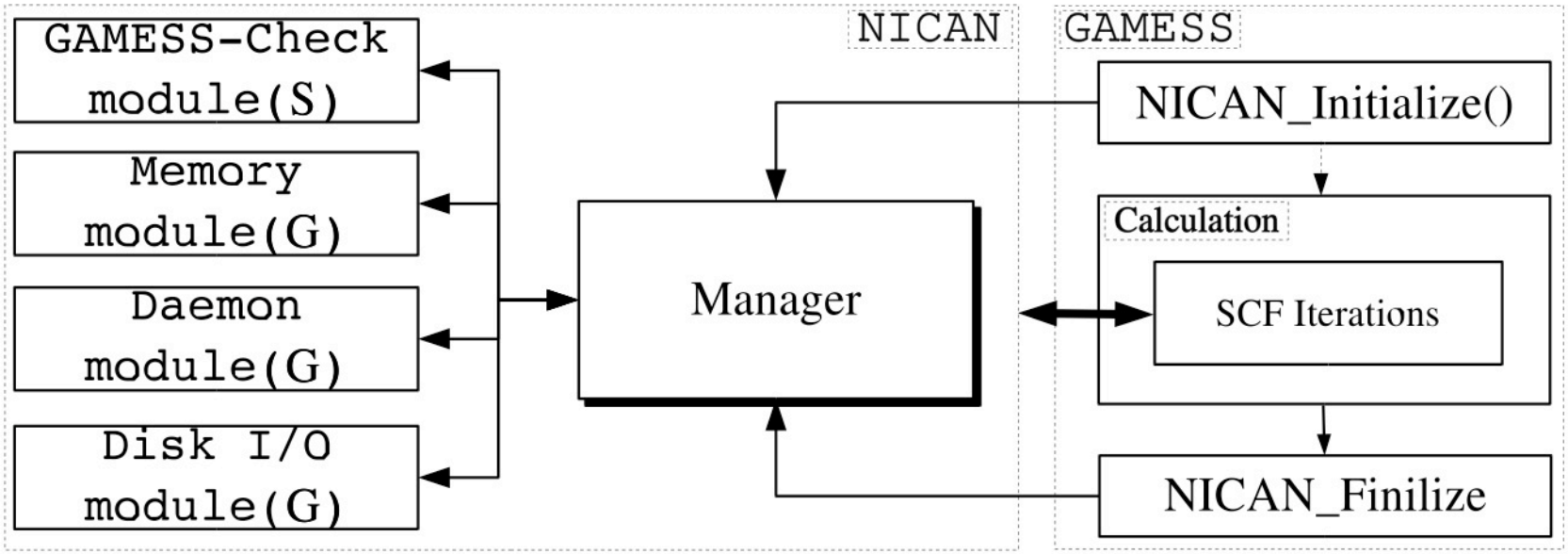
- Network Information Conveyer and Application Notification
- Decouples process of analyzing system information from application execution
- Enables adaptation functionality for distributed applications
- Requires minor changes to adapting application
- Lightweight module-driven middleware
  - CPULoad, Latency, PacketProbe, etc.

Introduction
NICAN



Introduction

GAMESS-NICAN Integration model



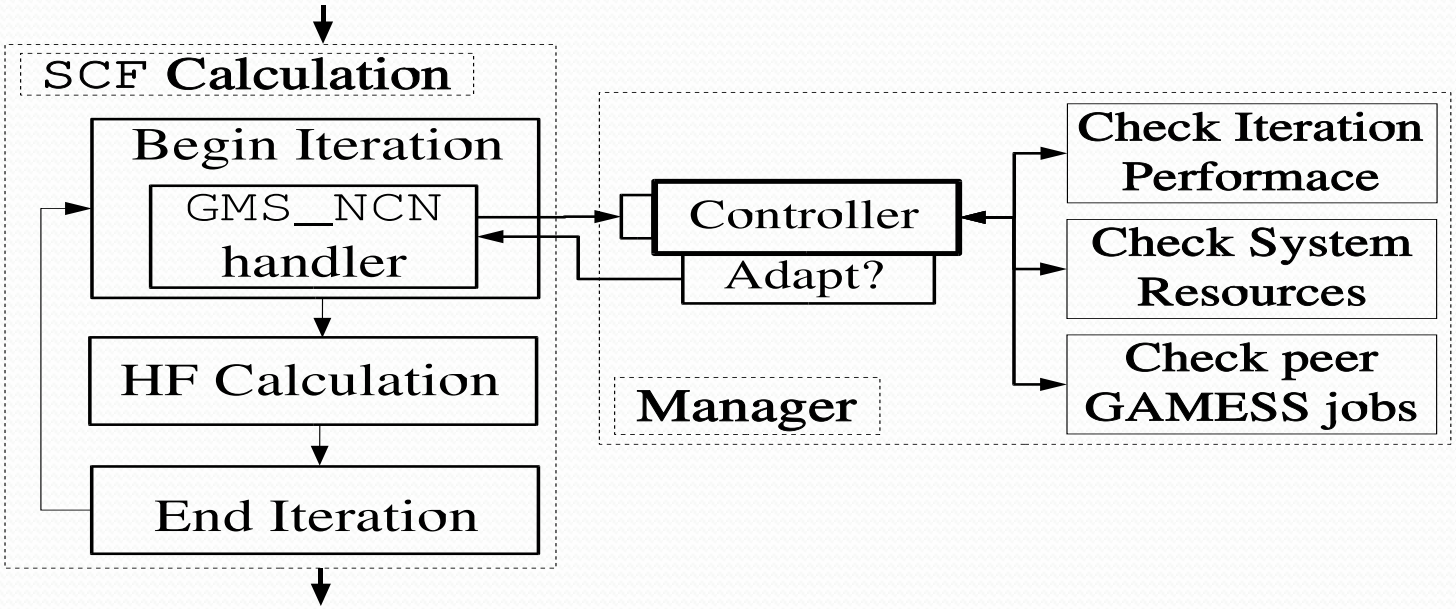
Introduction

Dynamic Algorithm Selection

- Assumes real-world scenario: GAMESS calculations are run in multi-user/application environment
- Examples: Disk I/O congestion may appear when an external application runs on the same SMP node as GAMESS
- Highlight of decision making process
  - Collect data
  - Compare current iteration performance to past and make decision
  - Switch algorithm

Introduction

Adaptation Process

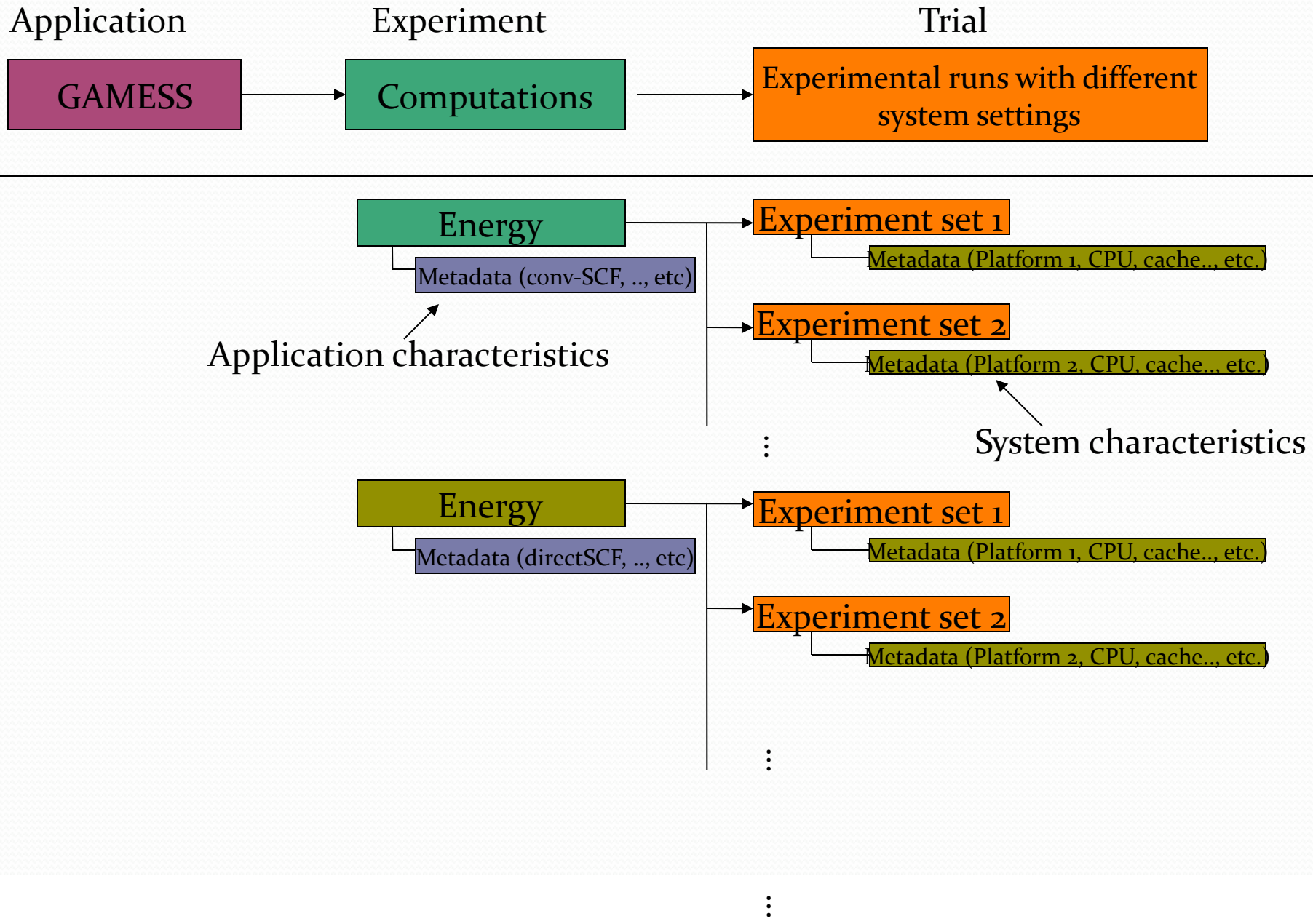


- Very few lines of GAMESS code change
- Low overhead by Manager

Reason to modify this adaptation scheme

- Algorithm effective in improving performance of GAMESS
- Iteration time data collected on-the-fly
- Need to include other parameters in the adaptation algorithm in order to reflect various scenarios that affect the application
- Hence collect application performance data on different architectures and then augment the existing adaptation scheme.

Methodology



Methodology

Application Workload

- Choose application workload to include different sets of molecules.
  - Molecules need to represent real world usage.
  - Two different sets of molecules chosen for testing
  - First set (Hiro molecules) of 7 molecules of varying molecular structure
  - Second set of 6 benzene molecules with very similar structure
  - Molecules represent fundamental aromatic systems, models used for DNA stacking and protein folding and are part of carbon nano materials.

Methodology
Architectures

- Choose different architectures on which the application can be tested.
  - Franklin : CRAY-XT cluster provided by NERSC
  - Sun T2 Niagara Machine: Single chip 8 cores. Each core capable of running 8 threads simultaneously.
  - Ames Lab SMP cluster “Borges" : 4 nodes. Each node contains two dual-core 2.0GHZ Xeon “Woodcrest" CPUs. Gigabit Ethernet interconnect between nodes.

Methodology

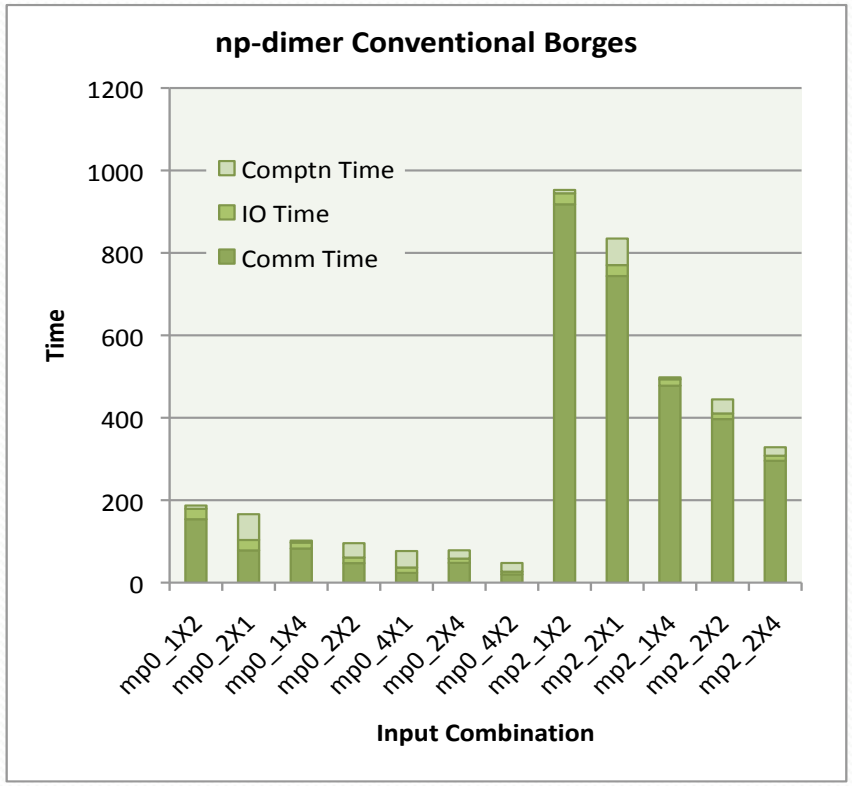
Performance Data and Tools

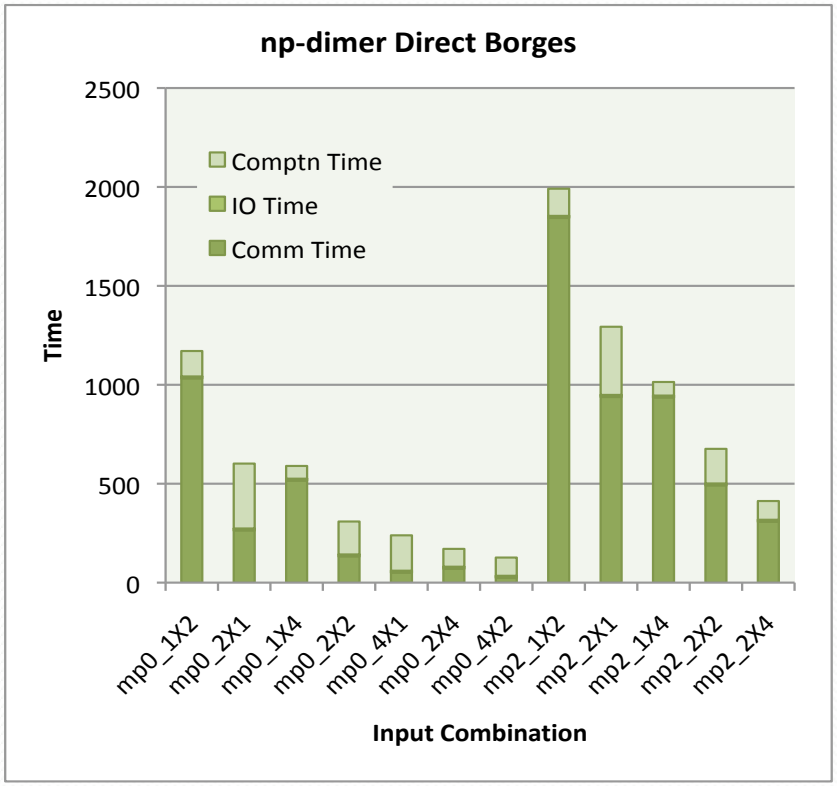
- Decide performance data to be collected
  - Overall time spent in Computation
  - Overall time spent in IO
  - Overall time spent in Communication
- Choose appropriate profiling tools to get the performance data.
  - TAU (Tuning and Analysis Utility)

Performance Analysis

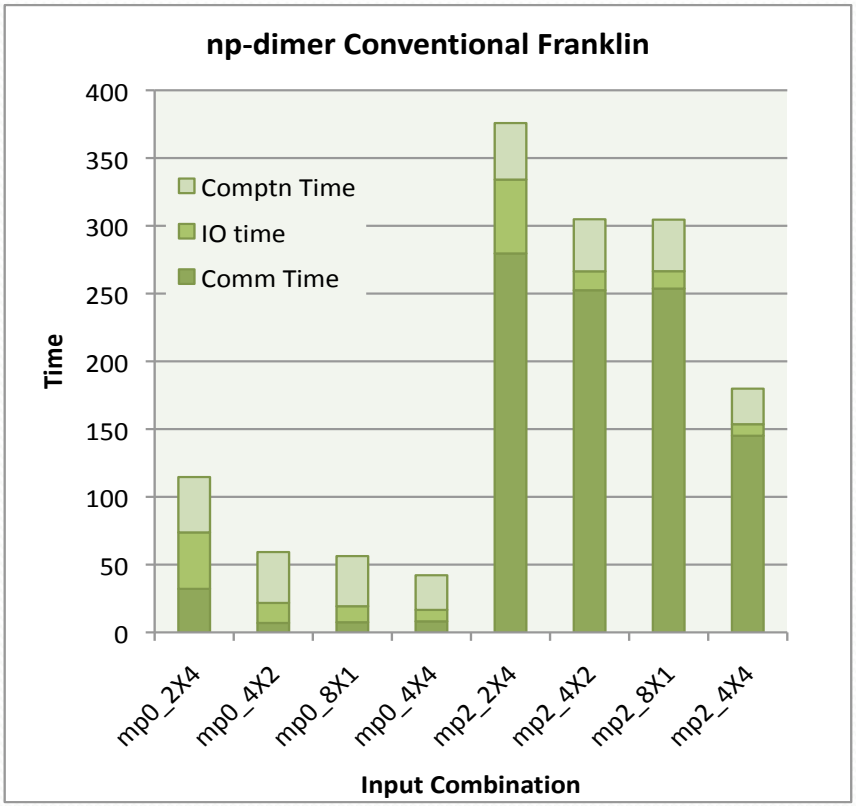
- Performance results shown only for np-dimer and C60 molecules.
- Results collected for input combinations of MP0, MP2, Direct and Conventional.

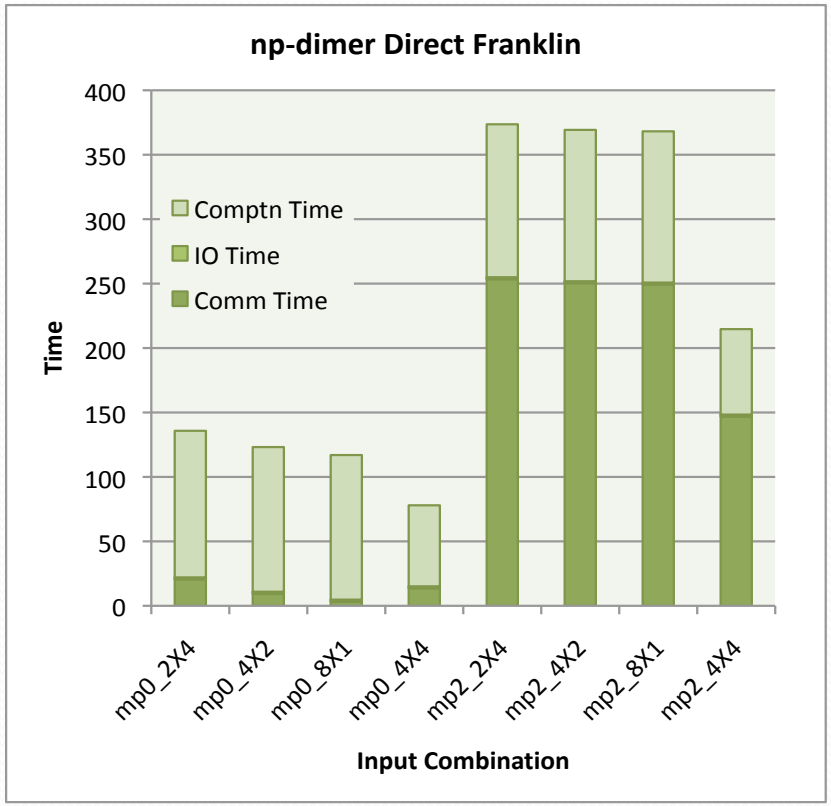
Performance Analysis
np-dimer Borges



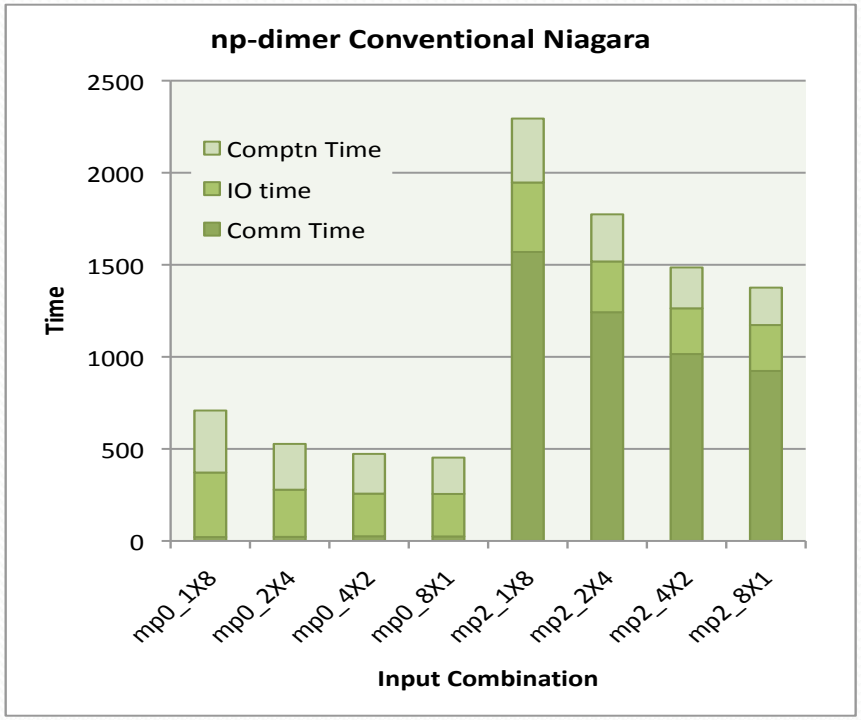


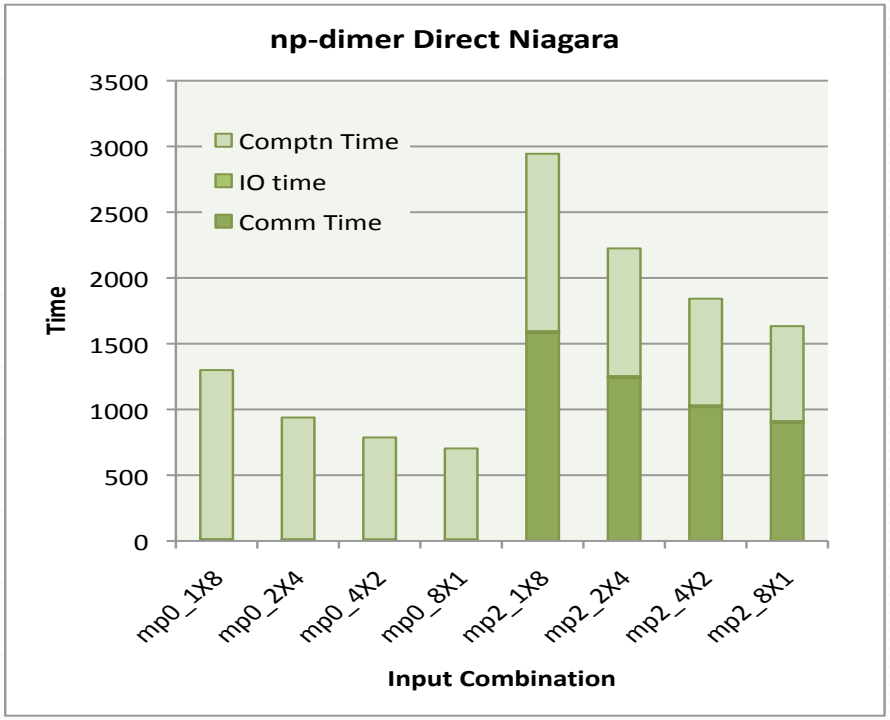
Performance Analysis
np-dimer Franklin



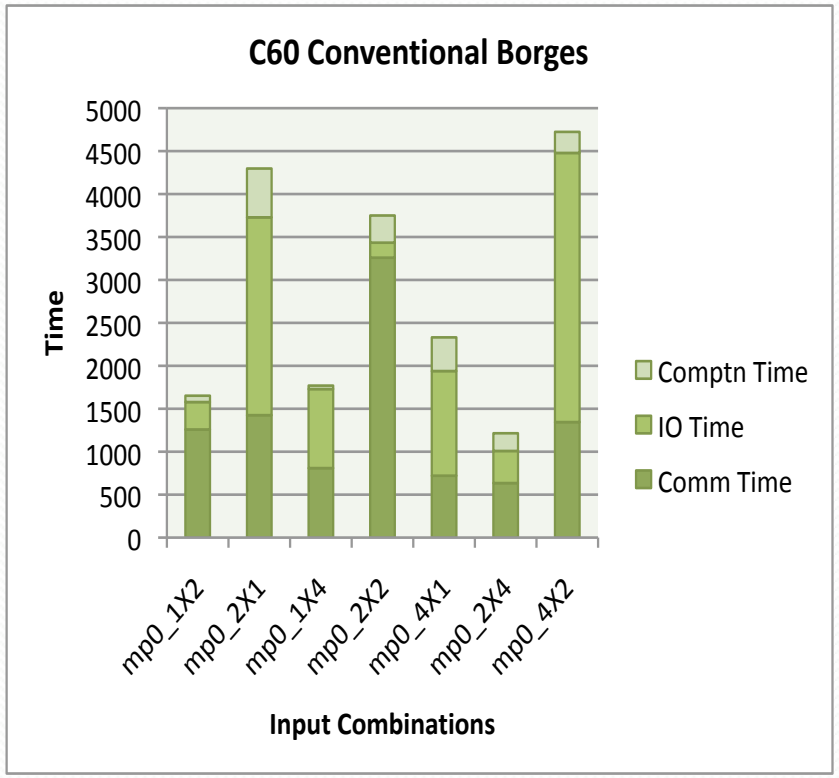


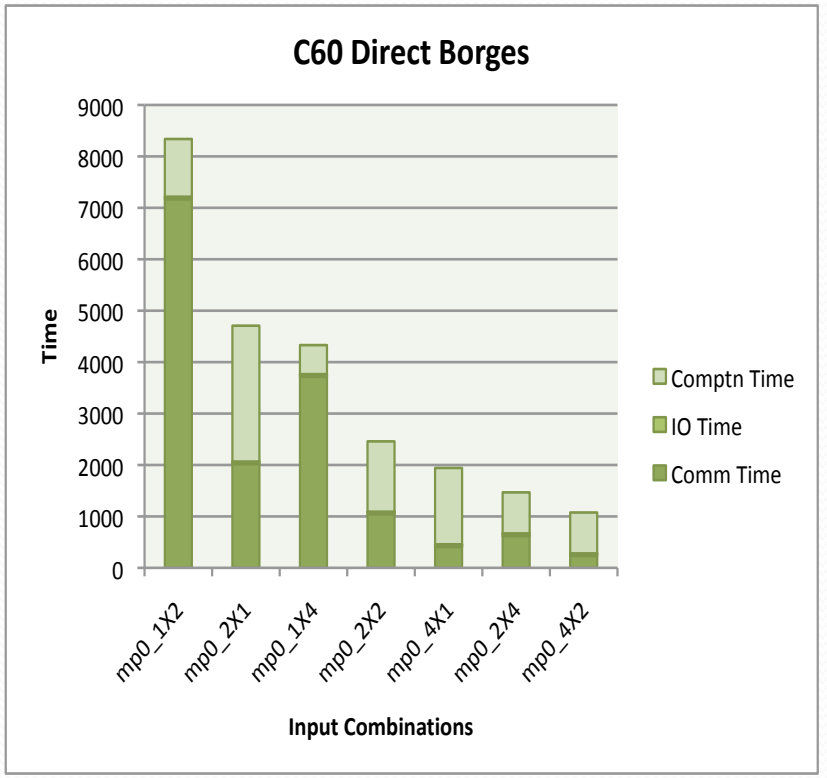
Performance Analysis
np-dimer Niagara T2



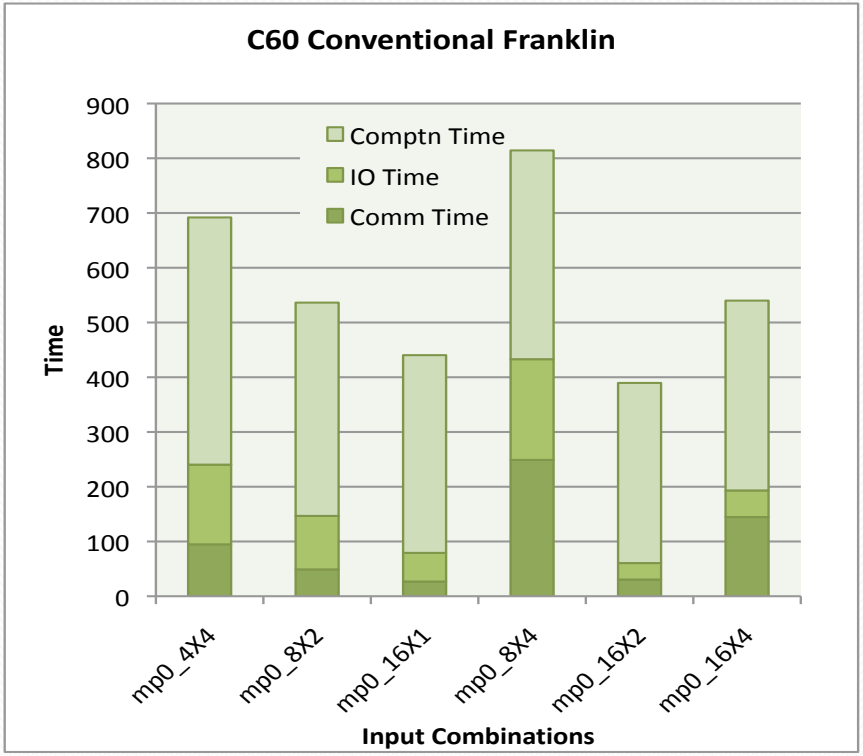


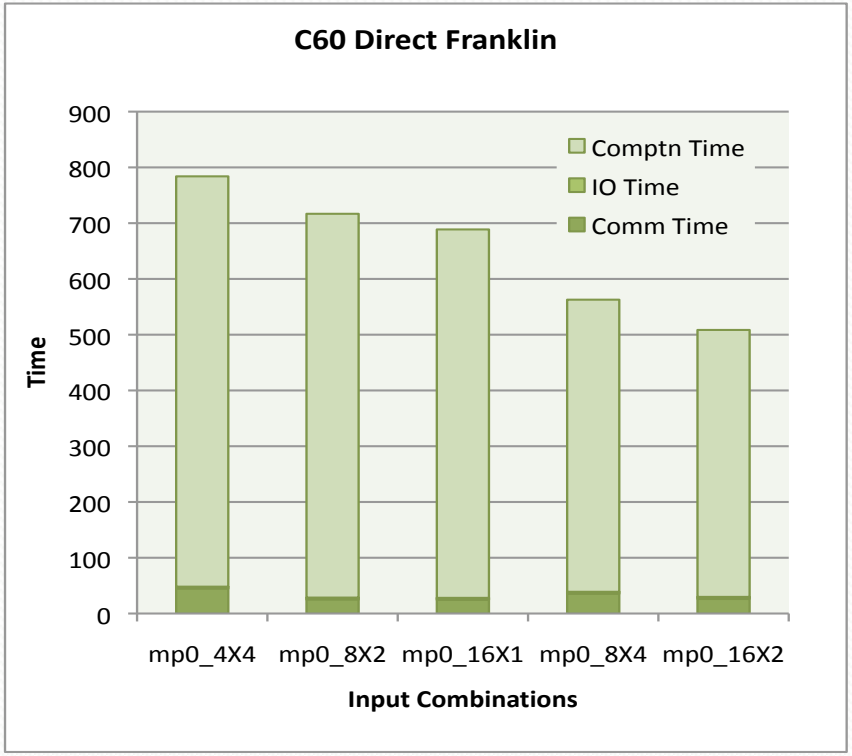
Performance Analysis
C60 Borges



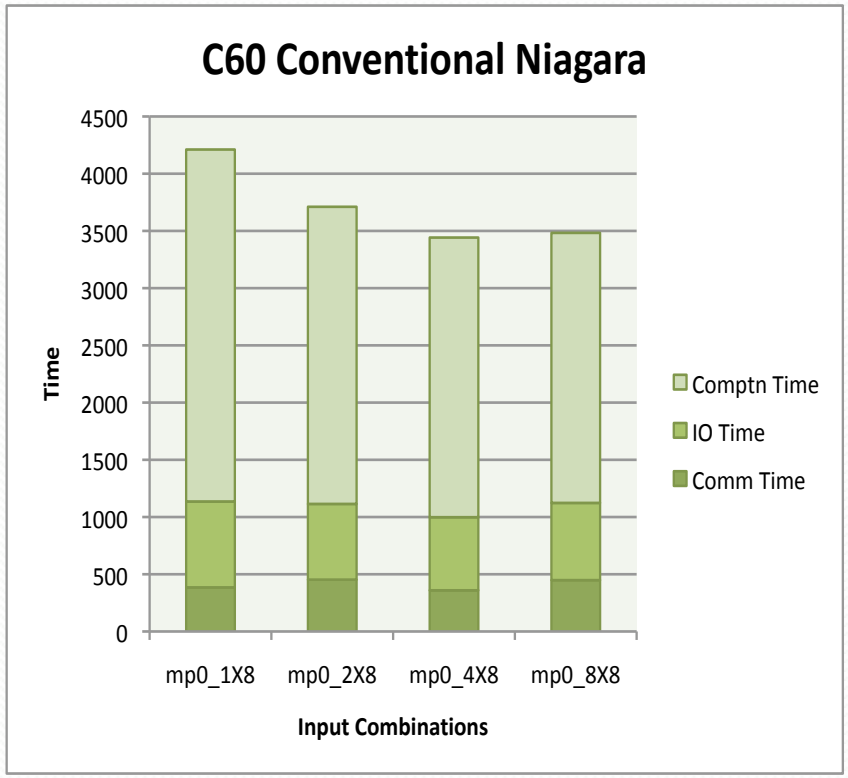


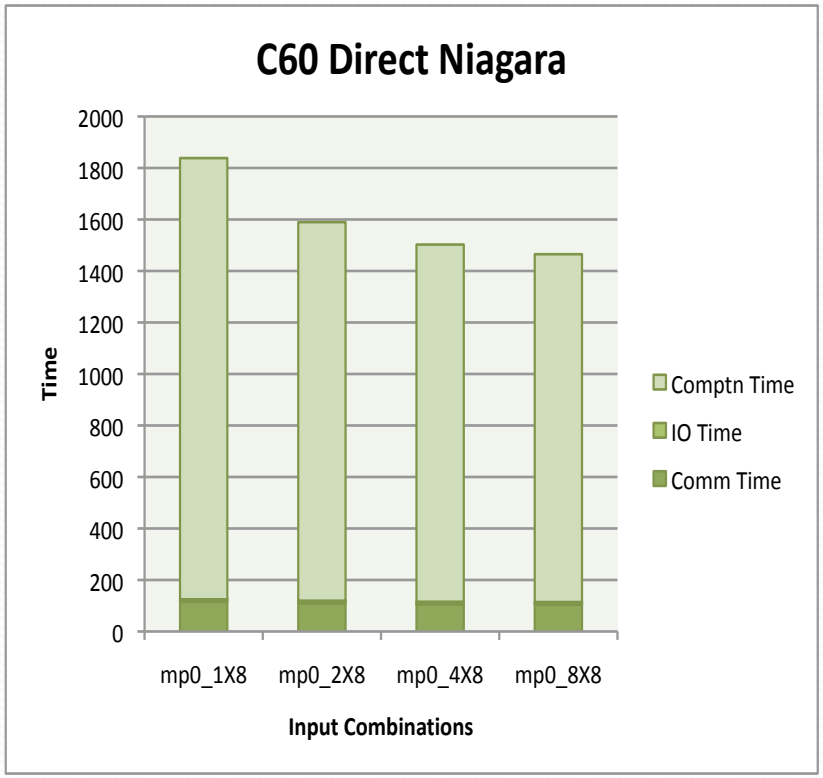
Performance Analysis
C60 Franklin





Performance Analysis
C60 T2 Niagara





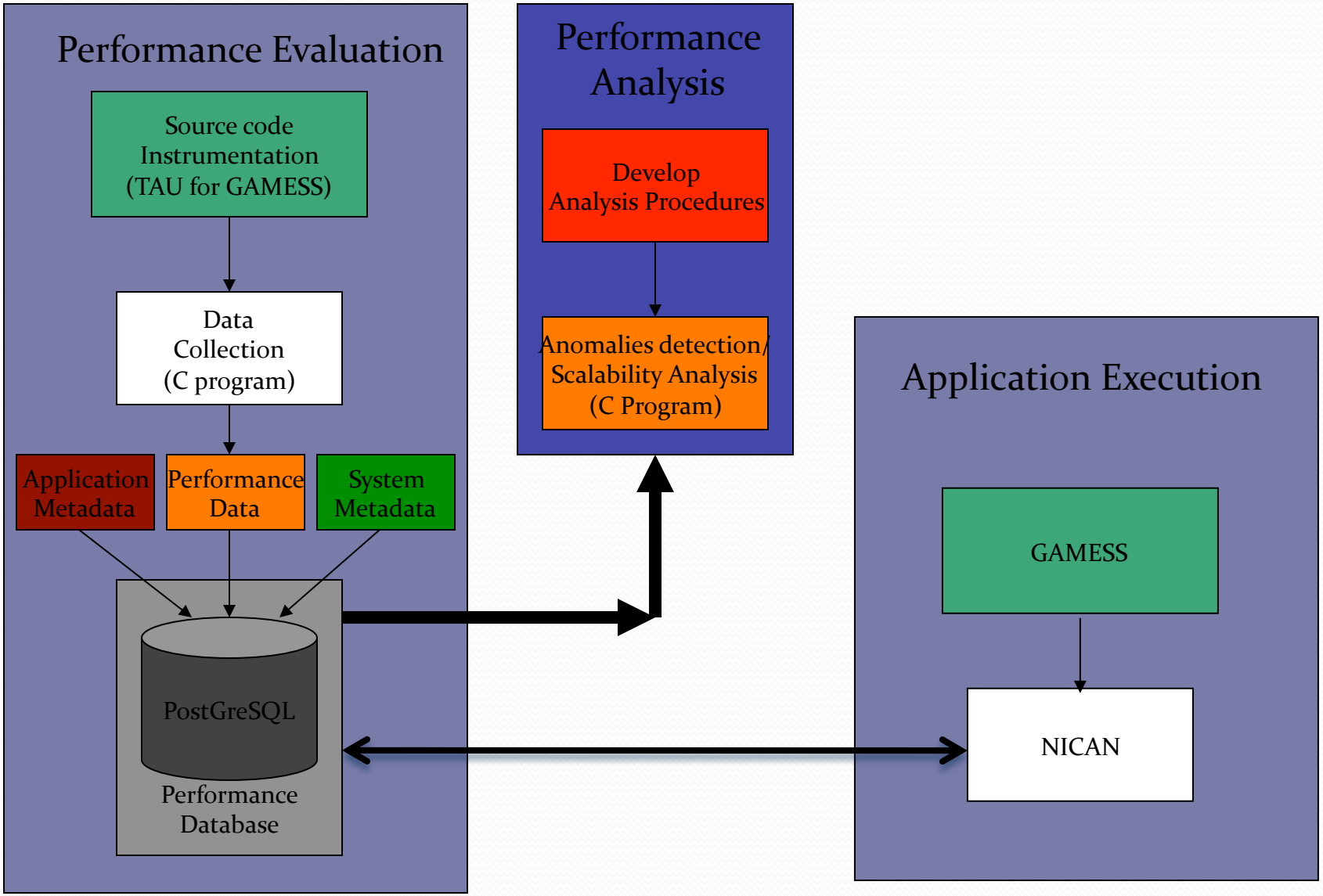
Issues in developing Tuning Strategy

- MP2 calculations take nearly 3 times more time to complete than MP0. There are other Post-HF computations. How can we make a trade off between accuracy and efficiency ?
- Communication cost increases when number of GAMESS processes on a single node is increased. Can we distribute the processes amongst different nodes ? How can the application know the best node-processor combination on a particular machine ?
- Are there input combinations that can be avoided based on the amount of time taken to compute results ?
- Can we use analysis results derived from one molecule for another ?

Issues in developing tuning strategy

- For a single molecule like np-dimer, for 4 different input parameter combinations, we obtained performance data on 3 architectures for at least 8 different node-processor combinations.
- 96 performance data sets for a single molecule.
- Need to store this data in a database for analysis.
- Dimension reduction needed for usage with NICAN

Database assisted adaptation architecture



Features implemented

- Memory usage check for MP2 computations
- Modification of input processor-node combination for better performance.
- Scalability analysis program implemented
- Improvement of about 8-9% over the existing NICAN implementation.

Conclusions and Future Work

- Huge amounts of performance data must be processed and organized.
- More detailed performance data can be used. Example: We can get Computation time, IO time and Communication time for specific execution phases.
- Other performance data like cache performance data can be added to the database and integrated with the tuning mechanism.
- Other scenarios need to be added to the tuning mechanism.
- Need to integrate tools like PerfDMF and PerfExplorer to manage and analyse the performance data.
- Use analysis techniques like machine learning.

Questions