High Performance Computing in LIA - UFC: Current Status and Future Directions

João Marcelo Uchôa de Alencar

4th Workshop of HOSCAR Project

15-19 September, 2014
Gramado, RS, Brazil
Agenda

• About UFC
• Activities developed by UFC Team
  – GREat
  – ParGO
UFC

- **Ceará**
  - Brazilian northeast state
  - 9 millions inhabitants
  - GDP: R$ 87,982 billions

- **Federal University of Ceará**
  - Founded in 1954
  - 42,443 students enrolled
  - 8 campuses across the state (Fortaleza, Quixadá, Sobral, ...)
  - Recently elected the 13th best university in Brazil by *Folha de São Paulo*
  - Patent applications increased 766% from 2008-2009 to 2010-2011
UFC Team involved in HOSCAR

- **GREat**
  - Prof. Dra. Rossana M. C. Andrade
  - Prof. Dr. José Neuman de Souza
  - Prof. Dr. Fernando A. Mota Trinta
  - Prof. Dr. Daniele Gomes
  - Prof. Dr. Miguel Franklin
  - Prof. Emanuel Ferreira Coutinho
  - Dra. Carina Teixeira de Oliveira
  - Ronaldo Lima
  - Felipe Anderson Maciel
  - Philipp B. Costa
  - Deborah Maria Vieira Magalhães
  - Prof. Paulo A. Leal Rego.
  - Jefferson Ribeiro
  - Renato Neto
  - Igor do Carmo
  - Samuel Soares

- **ARIDA**
  - José Antônio Fernandes de Macedo
  - Vinicius Pires

- **ParGO**
  - Prof. Dr. Heron de Carvalho
  - Prof. João Marcelo Uchôa Alencar
  - Prof. Jefferson Silva
  - Cenez Rezende
  - Wagner Al-Alan
  - Anderson Boettge
  - Neemias Gabriel
UFC

- LIA – UFC
  - Is the global lab of the Computer Science Department
  - www.lia.ufc.br

- GREat
  - The Group of Computer Networks, Software Engineering, and Systems
  - www.great.ufc.br

- ParGO
  - Parallelism, Graphs and Optimization
  - www.lia.ufc.br/~pargo

- CENAPAD-UFC
  - National Center for Supercomputing – UFC
  - www.cenapad.ufc.br, @cenapadufc
UFC

- LIA-UFC
  - is part of GREat
  - is part of ParGO
  - is a user of CENAPAD-UFC
CENAPAD-UFC

• Mission
  – To provide on-demand High Performance Computing (HPC) services to universities, research institutes and other public or private institutions
  – It is a national center that focuses on meeting the needs of research groups in the north and northeast.
  – SINAPAD

• Computational Cluster
  – Cluster Bull
    • 48 nodes, each with 12 cores e 24 GB RAM
    • Total: **576 cores e 1152 GB RAM**
  – GPUs Nvidia
    • 3 nodes, each also with 16 CPU cores e 96 GB RAM
    • 6 k20 boards
  – Storage
    • 145 TB

• Computational Cloud
  – 5 nodes:
    • Intel Core 2 Duo 2 cores and 2 GB RAM (controller)
    • 2x Intel Core i7 8 cores and 8 GB RAM
    • Intel Xeon com 4 cores and 16 GB RAM
    • Intel Xeon com 12 cores and 32 GB RAM
  – Total: **34 cores and 66 GB RAM**
  – Toolkit: OpenNebula
GREat

• Improving end user interaction with HPC resources
• Investigating future scenarios for Cloud Computing
  – HPC
  – Cloud Infrastructure Resource Allocation
  – Future Directions
    • Mobility
    • Security
• $a^2c$ – a web portal access to cluster

• Motivations
  – Enable easy access to a variety of users with different needs
  – Provide abstractions while retaining flexibility
  – Work as an entry point where scheduling policies may be applied without changing the underlying infrastructure

• INCT-MACC
a²c - Environment Overview
• **Portals**
  - **Generic**: graphical interface to SLURM (resource manager)
  - **NS3**: network simulation tool
  - **NAMD**: molecular simulation tool
Parâmetros da simulação

- Tamanho horizontal da grade (x-size)
- Tamanho vertical da grade (y-size)
- Espaço entre nós (step)
- Tempo de simulação (segundos)
- Número de interfaces de rádio por nó
- Intervalo de tempo entre transmissão pacotes (segundos)
- Tamanho dos pacotes (Bytes)
- Política de escolha de canais (channels)
- Tipos de trace desejado

Simulações

<table>
<thead>
<tr>
<th>Início da Simulação</th>
<th>status</th>
<th>Programa</th>
<th>Parametros da simulação</th>
<th>Baixar</th>
<th>Excluir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Não foi localizado nenhum arquivo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scientific Portals Module

Authentication Module

Decision Module
- Decision Algorithms
- User Permission
- Infrastructure Management

Environment Module
- Cluster Routines
- Cloud Routines

a²c – Architectural Overview
Cloud for HPC

- Cluster and Cloud Integration
  - Deploy cloud infrastructure at CENAPAD-UFC
  - Enhance user experience with cloud resources
  - If you have a powerful cluster, why do need a private cloud?
GREat

Cloud for HPC

• Cluster
  - Server class processors
  - Great memory per node
  - Fast interconnect
  - Low flexibility
    • Many nodes, changing the configuration is not trivial
    • There are devices with proprietary drivers that may not be updated
  - Expensive to expand
    • Hardware is available, but is expensive
    • Blade architecture may require new chassis
    • Infiniband expansion is not cheap
  - **Perfect tool for scientific computing**
Cloud for HPC

- Private Cloud
  - Server class and desktop class processors
  - Less memory per core
  - Ethernet
  - High flexibility
    - Virtualization allow different OS images to run in the same resource
    - Virtual machines may be updated easily without changing the physical host
  - Cheap expansion
    - Commodity hardware
    - Increasing the number of nodes is straightforward
- **Not the best performance for scientific computing**
Cloud for HPC

- Perfect scenario for the Cluster
  - A researcher wants to run parallel distributed applications with MPI
    - Low latency demand
    - Using the cloud would offer performance decrease due to ethernet
GREat

Cloud for HPC

• Possible scenarios for the Cloud
  – A researcher wants to run legacy applications or code developed by himself/herself with **specific requirements for number of processes, operating system, compilers, libraries**, etc
    • Changing the cluster setup may not be simple. For example, **incompatible** libraries versions
    • With virtualization, it is possible to create a software environment **identical** to the researcher's setting
  – A researcher wants to execute serial code or multithread only, without MPI
    • He/she is using CENAPAD for performance, but also for **reliability** (no-break, redundant power, etc)
    • Using the cluster may take nodes that would be better used by MPI applications
    • Running serial or multithread only code on the cloud offers **acceptable performance**
GREat

Cloud for HPC

• $a^2c$ Decision Module
  – If cluster usage is **high**, but still with available nodes
    • Send all new serial or multithread only jobs to the cloud
  – If cluster queue is **full**
    • Send all jobs (MPI or not) to the cloud. However, if available nodes appear on the cluster, migrate MPI applications from the cloud to the cluster. Migration must be supported by application (for example, GROMACS)
  – If cluster usage is **low**
    • Send all jobs to the cluster
Cloud for HPC

- **a²c Cluster and Cloud Integration**
  - It is still a work in progress, but from the preliminary data we can see that the queuing time is reduced
  - With more usage data, we expect to show that the overall execution time is lower
  - **Conclusion:** the cluster is faster, but the cloud is easier and cheaper to expand and use, may increase the job throughput and decrease configuration time before running applications

- **Future Work:**
  - Further study the execution of MPI applications on the cloud
  - Migration Strategies
  - Create more portals according CENAPAD-UFC's users needs
GREat

Cloud Infrastructure Resource Allocation

• Evaluation of software to set up a private/hybrid cloud
  - OpenNebula, OpenStack, Eucalyptus, CloudStack
• Creation of virtual appliances for easier the deployment of cloud applications
• Development of solutions to handle the heterogeneity of the data center’s physical machines to achieve an homogeneous performance (FairCPU architecture)
• Development of techniques to handle elasticity among different cloud solutions and cloud datacenters (hybrid cloud)
• Performance evaluation of parallel applications to Big Data
  - Hadoop, YARN
Development of solutions to improve the performance of mobile applications and reduce battery consumption
- Exploit cloud capabilities (storage and compute) through the use of offloading techniques
- Orchestration of cloud services in private/local resources (cloudlet concept) and public/remote resources
- Frameworks for Android and Windows Phone

Study to improve the performance of private cloud infrastructure for different workload behavior of mobile applications

Handle mobility issues of such kind of application
- Handoff, loss of connectivity, mobile applications

Handle QoS and SLA for mobile applications
Cloud Security

• Data stored on the public cloud should be kept private
• The security requirements might be different
  – The SLA negotiation should regard the customers needs
  – The provider might cash in accordance with the defined security level
• SLA Violation
  – The customers should identify if the SLA was violated
  – The provider should use mechanisms to avoid a violation or to repair after a violation
• How to assure the data privacy when they are stored or processed in the cloud?
  – The metrics related to the parameters should be measurable
  – The negotiation can be automated
GREat

Publications


ParGO

- Hash Component Model
- Hash Programming Environment (HPE)
- HPC STORM
Separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern.

- Philip Laplante - “What Every Engineer Should Know About Software Engineering”

A concern is a set of information that affects the code of a computer program.

A program build upon SoC is said to be modular.
For HPC codes, some examples of concerns

- A piece of code that represents some meaningful calculation, for example, a local matrix-vector multiplication
- A collective synchronization operation, which may be represented by a sequence of send/recv operations;
- A set of non-contiguous pieces of code including debugging code of the process;
- The identity of the processing unit where the process executes;
- The location of a process in a given process topology
Emerging large scale HPC applications from computational sciences and engineering
  - Software engineering requirements
  - Collaborative environments
  - Capability/capacity computing platforms
  - World-wide scale collaboration and computation

The **Hash Component Model** enables the development of Component-Based High Performance Computing (CBHPC) applications
  - The separation of **concerns** through process slicing
  - **Orthogonality** between processes and concerns as units of software decomposition
Let $A$ and $B$ be $n \times n$ matrices and $X$ and $Y$ be vectors. It computes $(A \times X) \cdot (B \times Y)$.

**Hash Example:** Slicing a simple parallel program by concerns.
Hash Example: Slicing a simple parallel program by concerns
A component model for distributed-memory parallel programs

Units + overlapping composition + component kinds
A component model for distributed-memory parallel programs

Units + overlapping composition + component kinds
ParGO
Hash Component Model

- A component model for distributed-memory parallel programs
- Units + overlapping composition + component kinds
ParGO

Hash Component Model

- A component model for distributed-memory parallel programs
- Units + overlapping composition + component kinds
ParGO
Hash Programming Environment (HPE)

• A reference implementation of the Hash Component Model
  - Focus on cluster computing platforms

• Architecture: Front-End, Core and Back-End

• From the Front-End, programmers build new components by composition of component contracts retrieved form the Core, register them into the Core, and run applications in a parallel computing platform through the Back-End service;

• The Core is a component catalog, with tuned implementations for different application and execution platform contexts;

• When running an application, the Back-End looks at the Core for the best implementation of a parallel component for the architecture of the parallel computing platform it represents.
Thee distinct architectures of parallel computing platforms

HPE Conceptual View
Thee distinct architectures of parallel computing platforms

Front-End

Core

Back-End

component catalog

provide components (2)

resolve contracts (3)

application

execute application (1)

component contracts

HPE Conceptual View
How to define components (contracts) that specify two things:
- The concern to be addressed
- The implementation assumptions about the execution context
  - execution context = parallel computing platform + application
  - goal: select the best component for each context

For component reuse, the programmer details the concern and the contextual parameters (Abstract Component)
- HPE finds the closest concrete component available (actual code)
ParGO

Hash Programming Environment (HPE)

**LINEAR_SYSTEM_SOLVER**

\[
[\text{accelerator\_type} = A: \text{ACCELERATOR\_TYPE}, \text{multicore\_support} = M: \text{MULTICORE\_SUPPORT}, \text{matrix\_pattern} = P: \text{MATRIX\_PATTERN}, \text{matrix\_partition} = R: \text{MATRIX\_PARTITION} [\text{multicore} = M], \text{matrix\_type} = T: \text{MATRIX\_TYPE} [\text{property} = P, \text{partition} = R]]
\]

An abstract component signature with context parameters.
ParGO

Hash Programming Environment (HPE)

• Evaluation
  – Implementing the NAS Parallel Benchmark (NPB) on HPE
  – Programs implemented:
    • FT, LU, SP and BT
  – Problem Classes
    • W and A
  – Comparison between the Fortran code version translated to C# and a Component-Based version
  – Castanhão Cluster
    • 16 nodes with 2 Intel Xeon 1.8 Processor
    • 32 GB RAM Total
    • Gigabit Ethernet
    • GCC Compiler
    • Mono 2.4
Decomposing SP and BT in Components
ParGO
Hash Programming Environment (HPE)

SP - W

C - min
C - max
M - min
M - max
C - avg
M - avg
Sequential

t (s)
Processes
ParGO
Hash Programming Environment (HPE)

SP - A
Conclusions

- The Hash Component Model provides a better way for code organization
- Overhead due to a component-based architecture may be negligible

Future Work

- Cloud architectures (see next...)
- Modeling other HPC applications:
  - Map-Reduce Algorithms
  - Graph Algorithms
HPC Storm is

HPC in clouds through components

HPC applications
- computational sciences
- engineering
- ...

ParGO
HPC Storm

components

CBHPC Hash

CBSE in clouds

HPC in clouds

HPC

clouds

HPC Storm
• **Services**
  - **IaaS (infrastructure):** comprising parallel computing platforms
  - **PaaS (platform):** for developing components and applications that may exploit the potential performance of these parallel computing platforms
  - **SaaS (software):** built from components, for attending HPC users

• **Stakeholders**
  - Domain **Specialists**
  - Application **Providers**
  - Component **Developers**
  - Platform **Maintainers**

• **Architecture**
  - Front-End, Core, Back-End
Front-End (SaaS)

providers

developers

maintainers

specialists (final users)

use applications

Core (PaaS)

build applications

build components

manage infrastructure

components

Back-End (IaaS)

includes

parallel computing platforms

build components

built from

applications

include
• **Current Status**
  - A new **Core** enhanced with ontological resource description for clouds components
    - Phd Student: Wagner Al-Alam
• **Ongoing Work**
  - A redesigned **Front-End** for the cloud, with support for Workflows, Domain-Specific Languages, etc
    - Phd Student: Jefferson Carvalho
  - **Back-End** with support for component adaptation with Elasticity, scale-out/scale-in virtual nodes
    - Phd Student: João Marcelo
ParGO

Publications


Thank you!