OddCI: On-Demand Distributed Computing Infrastructure

Rostand Costa
Francisco Brasileiro
Guido Lemos Filho
Dênio Mariz Sousa

MTAGS
2nd Workshop on Many-Task Computing on Grids and Supercomputers

Co-located with
ACM/IEEE SC09 (International Conference for High Performance, Networking, Storage and Analysis)
Portland, Oregon – November 16th, 2009
Agenda

- Motivation
- DCI requirements for MTC
- OddCI: a novel approach to DCI
- OddCI over a digital TV network
- Performance assessment
- Concluding remarks
Introduction

- MTC speeds up execution of applications, but...
  - Large amount of parallelism can only be achieved if there is a relatively high level of independency among the sub-tasks
  - The scheduler need to have access to a huge number of processors.

- In this paper we are concerned with the issue of
  - Providing ways to assemble large pools of processors for the execution of MTC applications.
  - In particular, we focus on large-scale distributed computing infrastructures (DCI)
The throughput achieved by MTC over a DCI depends on the scale it allows.

To provide extremely high-throughput computing to a large number of applications, a DCI must meet some requirements:

- **extremely high scalability**: it must be able to handle up to hundreds of millions of processing resources in the same way that it handles a few dozens of them;
- **on-demand instantiation**: it must offer mechanisms for discovery, assemblage and coordination of the required resources, on demand and for a specified amount of time;
- **efficient setup**: the configuration of the processing nodes must be carried out quickly and demanding minimal interventions.
Available Alternatives

➡️ Desktop Grid Computing
  - the combination of computer resources from a single or multiple administrative domains applied to a common task
  - e.g. Condor, OurGrid, Alchemi

➡️ Voluntary Computing
  - a type of distributed computing in which computer owners donate their computing resources (such as processing power and storage) to one or more "projects“.
  - e.g. SETI@home, FightAIDS@home, Folding@Home

➡️ Infrastructure as a Service (IaaS)
  - the delivery of computer infrastructure (typically a platform virtualization environment) as a service
  - e.g. Amazon Elastic Compute Cloud (Amazon EC2),
No available technology is able to simultaneously address all the requirements to provide **extremely high-throughput** computing to over a DCI.

### Available Alternatives Vs Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Available Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Voluntary Computing</td>
</tr>
<tr>
<td><strong>Extremely High Scalability</strong></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Efficient Setup</strong></td>
<td>✗</td>
</tr>
<tr>
<td><strong>On-demand Instantiation</strong></td>
<td>✗</td>
</tr>
</tbody>
</table>
On-Demand Distributed Computing Infrastructure

Consider a special category of devices which may be organized as a broadcast network:

- Mobile phones, Digital TV receivers, Cable TV receivers
- Devices connected to the Internet with reasonably powerful processors
- Broadcast network can access simultaneously all the devices which can be coordinated to run some task
On-Demand Distributed Computing Infrastructure

- A novel architecture for generic DCI
- Flexible
  - Can be used for several scenarios and with different technologies and devices
- Potentially highly scalable
  - Millions of potential devices
- On-demand instantiation
  - Resources are discovered and allocated as required and for a specified amount of time
- Efficient setup
  - Building DCI instances with millions or thousand nodes demands similar effort via broadcast communication
OddCI Architecture

- **Provider**: creates, manages, destroys OddCI instances
- **Controller**: Setup, controls, sends software images, monitors PNA status
- **Backend**: schedules tasks, provide input data, collects output data, post-processing
- **PN Agent**: actually runs tasks, processes control messages
OddCI Architecture: operation

User submits a “processing request” to the provider

- DCI instance size (number of processing nodes)
- Application image, common data
- Node requirements
OddCI Architecture: operation

- **Provider** evaluate the user request
  - checks availability
  - keeps control information
- **Command the Controller** for creating the OddCI required instance
OddCl Architecture: operation

- **Controller** triggers a wakeup process to **PNAs** through the broadcast channel
  - PNA can drop jobs when busy or accept when idle
- **Controller** also send other control messages (e.g. dismantle instances)
- All **PNA** receives messages simultaneously
OddCI Architecture: operation

- **PNA** loads application image for execution in a DVE (Dynamic Virtual Environment)
- **Controller** monitors active **PNA**
- **Direct channel** is a two-way road
  - Application can interact with the **Backend** for requesting specific input data or send results (optional)
  - **PNA** sends status messages frequently to the **Controller**
Proof of Concept: OddCI over a Digital TV Network

Why DTV network?

- Open technology, well-defined standards
- Native transmission of data in broadcast
- Fast expansion, being deployed in many countries
- Great spectrum of devices: from set-top boxes to mobile devices
- Potential for millions of devices
- Powerful middleware

And also ...

- Feasibility for building a testbed
- Previous experience of our group
DTV Generic Model

Content Production → Digital TV Broadcaster → Digital TV Receiver

- Content
- Broadcast Network (Air, Cable, Satellite)
- Audio, Video, Data
- IP Packets
- Return Channel (Internet)
- Applications & Data
Implementing OddCI over DTV components

- DTV Receiver
  - Application Xlet
  - PNA Xlet
  - Middleware

- Processing Nodes

- Broadcast Channel

- Controller
  - DTV Broadcast Transmission
    - DSM-CC
    - MPEG-2 Transport Stream

- Direct Channel
  - DTV Return path
    - Internet

- Backend

- Provider

- Controller Integration
  - Gateway
  - Carousel Generator
Experiment setup

Experiments were performed using:

- **SBTVD** (Brazilian DTV standard)
- **Software**
  - Brazilian middleware “Ginga” implementation from UFPB
  - NCBI Toolkit ported using a cross-compiler
  - BLAST – Basic Local Alignment Search Tool tasks, from NCBI
- **DTV Receiver**
  - STI microelectronic’s processor ST7109
  - 32MB Flash memory, 256MB RAM
- **Reference system**
  - Dual Core Pentium, 1.6GHz, 1GB RAM, Debian Linux
DTV STB Performance - Preliminary findings

Experiment Setup
- BLAST application running in a STB and compared with a reference PC desktop
- STB with Brazilian middleware “Ginga”
- Tests performed using the cheapest STB in the Brazilian market (~US$ 100)

Remarks
- Ref PC is ~31 times faster than STB “in use” mode
- Ref PC is ~17 times faster than STB “standby” mode
Performance Assessment

→ Simple analytical model was developed

→ System parameters
  - Broadcast channel capacity = 1Mbps
  - Return channel capacity = 150kbps
  - Image size = 10Mb
  - Application Input + output data size = 1kb
  - Average Processing time = from 53ms to 1.5 hour
Research Roadmap

- Define an (ideally generic) architecture
- Preliminary analysis
  - STB basic performance assessment
- Proof of concept
  - Digital TV Network

- Complete analysis
  - Mathematical model, simulation
  - Case studies with different application profiles
- Security issues

- Optimization in specific components (e.g. controller)
- Business models
  - Voluntary computing model, reward model
  - Possibly with a real DTV broadcaster’s partnership
Concluding Remarks

OddCI: a novel approach to DCI
- Efficient setup, on-demand instantiation
- Great potential to enable DCI for Extremely High-Throughput Computing

OddCI can be instantiated over a DTV system
- Less processing power, but huge pool size
- Brazilian DTV expects ~100 million receivers by 2016
- European DVB: >500 million receivers deployed
- Chinese DTMB: ~100 million receivers (estimated)

Great research challenges to deal with
## OddCI: On-Demand Distributed Computing Infrastructure

### Contact Information

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rostand Costa</td>
<td><a href="mailto:rostand.costa@lsd.ufcg.edu.br">rostand.costa@lsd.ufcg.edu.br</a></td>
</tr>
<tr>
<td>Francisco Brasileiro</td>
<td><a href="mailto:fubica@lsd.ufcg.edu.br">fubica@lsd.ufcg.edu.br</a></td>
</tr>
<tr>
<td>Guido Lemos Filho</td>
<td><a href="mailto:guido@lavid.ufpb.br">guido@lavid.ufpb.br</a></td>
</tr>
<tr>
<td>Dênio Mariz Sousa</td>
<td><a href="mailto:denio@ifpb.edu.br">denio@ifpb.edu.br</a></td>
</tr>
</tbody>
</table>