Just in Time Clouds

Rostand Costa[1,2], **Francisco Brasileiro**[1]
Guido Lemos Filho[2], Dênio Mariz Sousa[2,3]

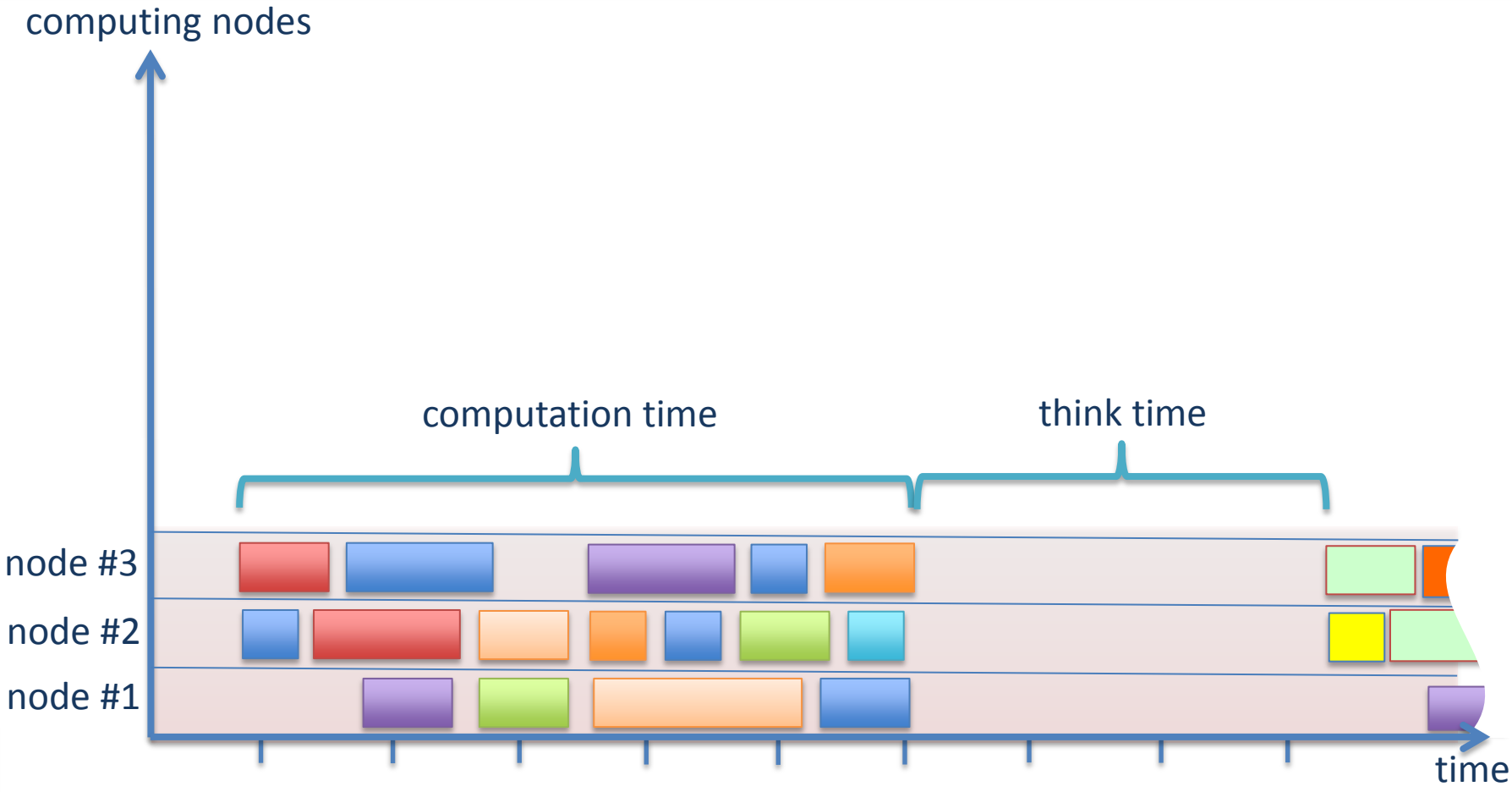
[1]Universidade Federal de Campina Grande
[2]Universidade Federal da Paraíba
[3]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

High Perfomance Scientific Computing in Clusters, Grid, and Cloud Computing Systems, Montivideo, November 9th, 2012

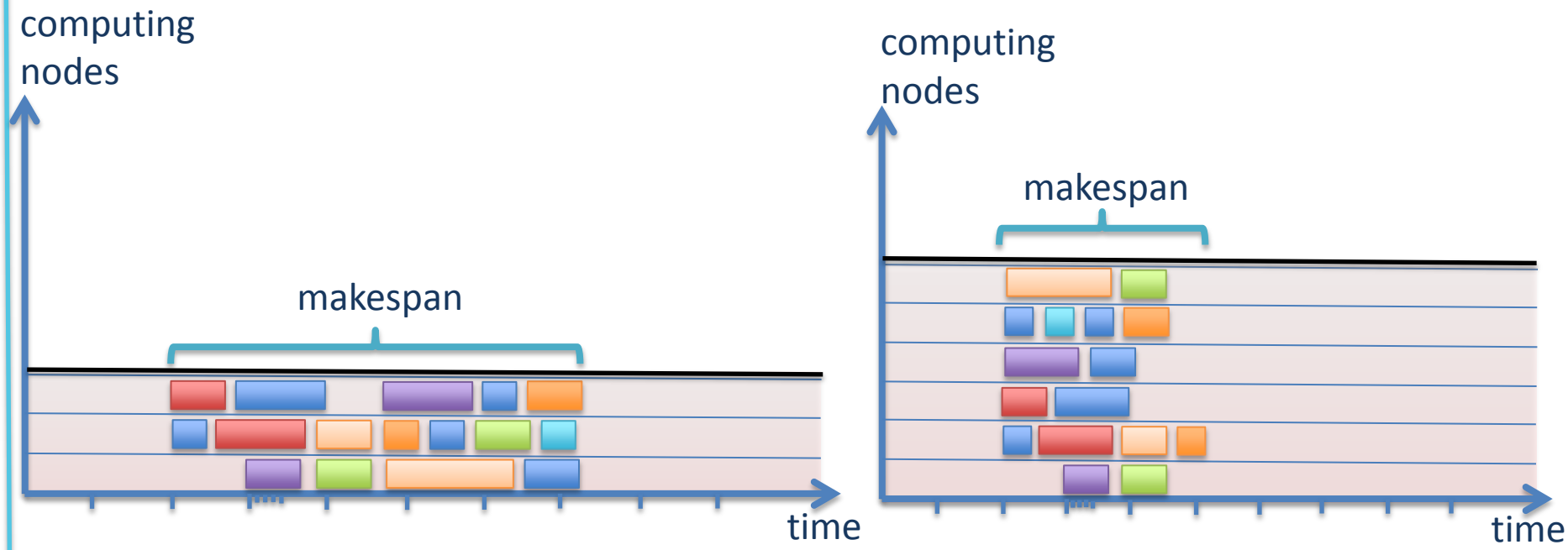
Universidade Federal de Campina Grande

Bag-of-task applications

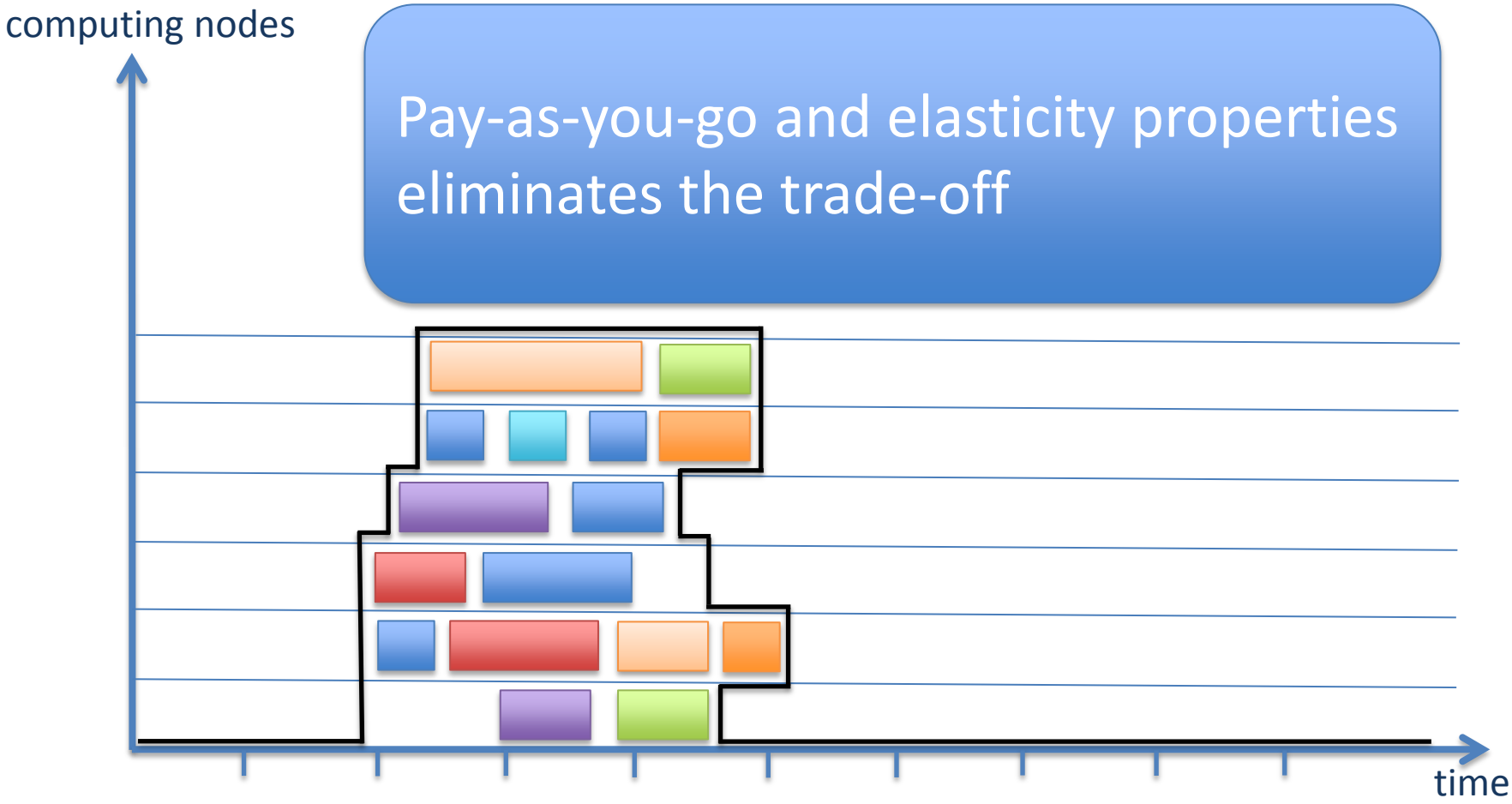
Typical workload of BoT applications



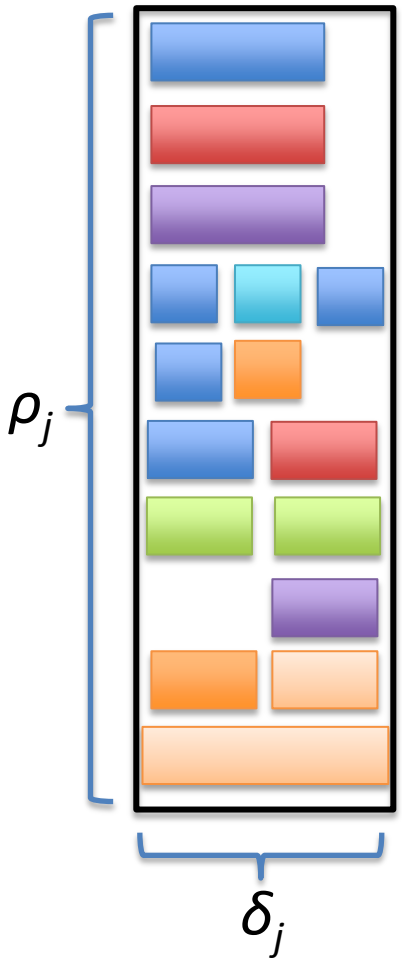
Trade-off performance vs idleness level



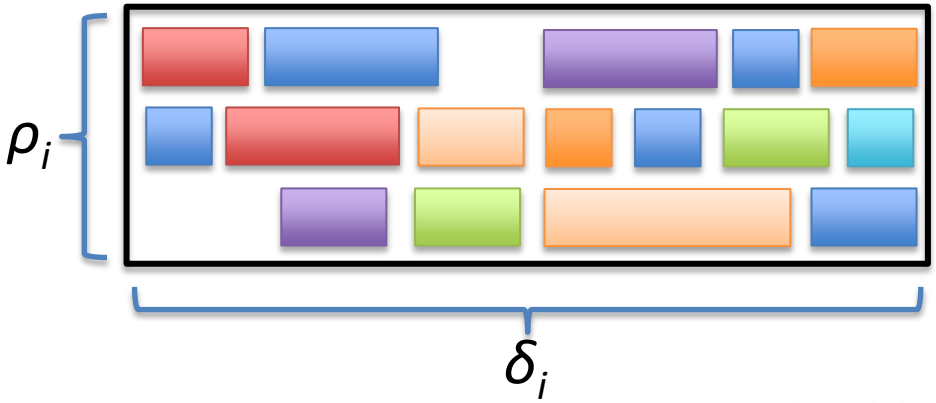
Running BoT on cloud computing IaaS providers



Cost associativity of cloud providers



Same charge for any two requests s_i and s_j, with $\rho_i \times \delta_i = \rho_j \times \delta_j$

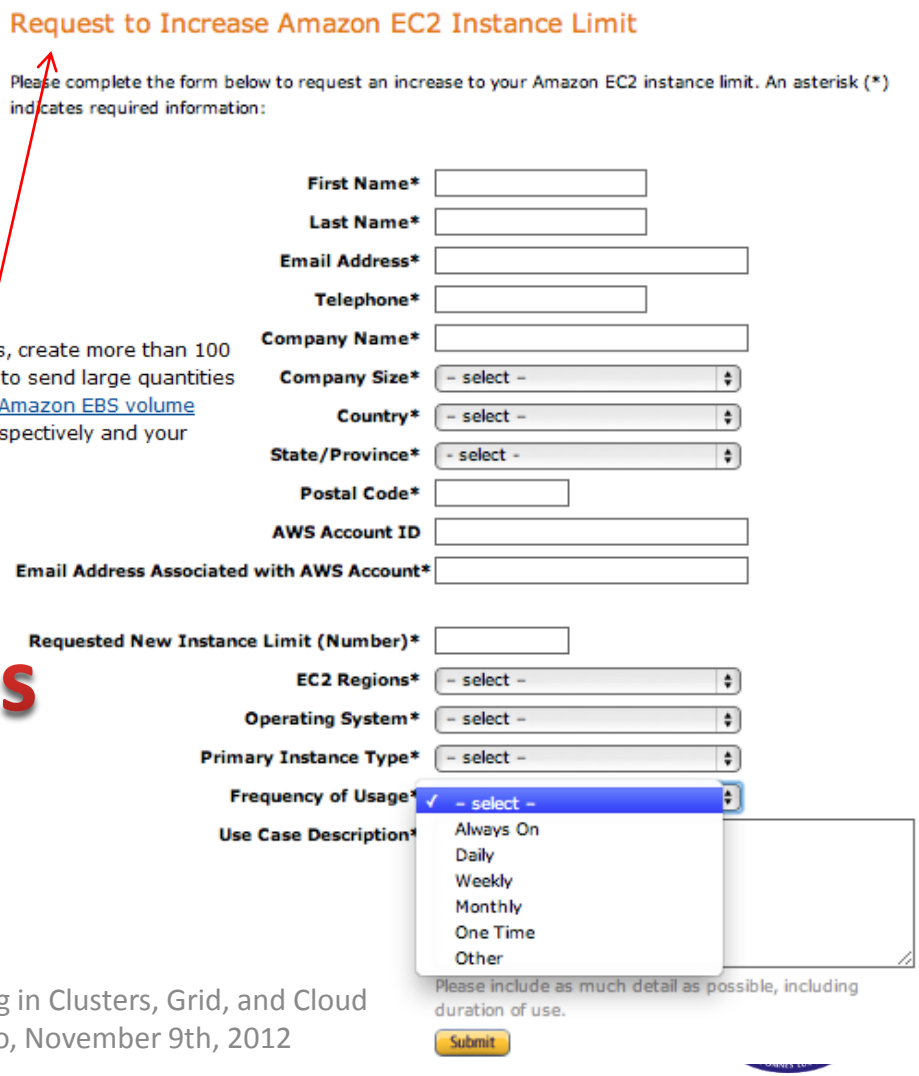


How elastic can one be in practice?

20 on-demand instances

If you wish to run more than 20 On-Demand or Reserved Instances or 100 Spot Instances, create more than 100 EBS volumes, need more than 5 Elastic IP addresses or 5 Elastic Load Balancers, or need to send large quantities of email from your EC2 account, please complete the Amazon EC2 instance request form, Amazon EBS volume request form, Elastic IP request form, Elastic Load Balancers, or the Email request form respectively and your request will be considered.

100 "spot" instances



This led us to our first research question …

- Why public cloud providers impose limits that restrict the usefulness of their services for clients with BoT applications?
  - How this limit affects the minimum capacity required by the provider?
  - How the rate of the infrastructure's idleness increases with the increase of this limit?
  - How extremely elastic usage patterns impact the cost for the provider to run its infrastructure?

Methodology

- Our approach is based on the use of simulation
- We defined a simplified model for IaaS providers …

  … and, in the absence of real workload traces, an appropriate synthetic workload generator for the proposed model
- We performed an experimental design to identify the random variables that had a major impact on the response variables of the model …

  … and, in the absence of data from cloud providers, we performed a parameter sweep over the main factors identified

Model sketch

- We follow a business-driven approach:
  - The provider's profit over a period of time ΔT is given by:
    - the revenue of the capacity sold during ΔT

      MINUS
    - the cost of maintaining the infrastructure during ΔT

      MINUS
    - the extra cost incurred only when its capacity is actually used during ΔT

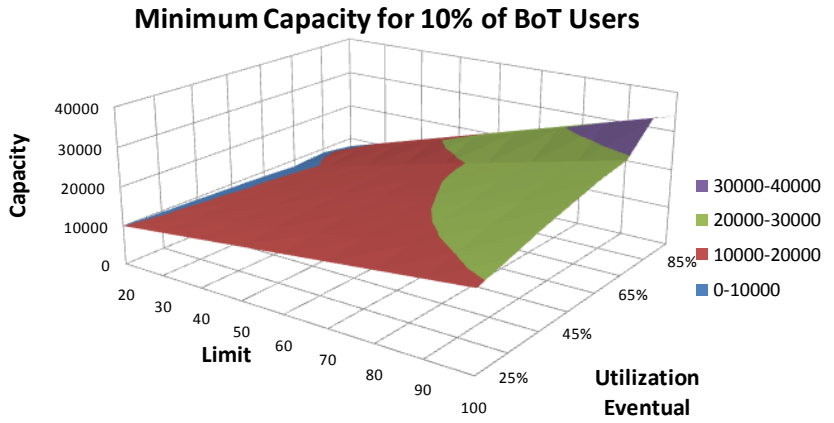
      MINUS
    - the cost of any violations incurred during ΔT (e.g. not being able to serve a client's request)

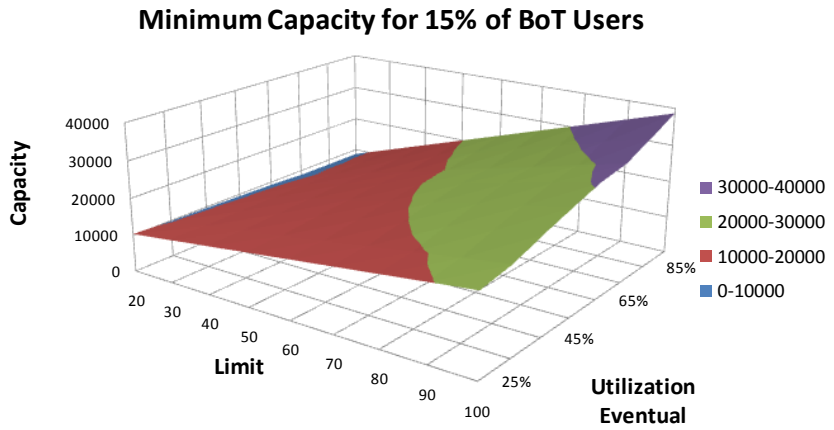
Synthetic workload: user-based model

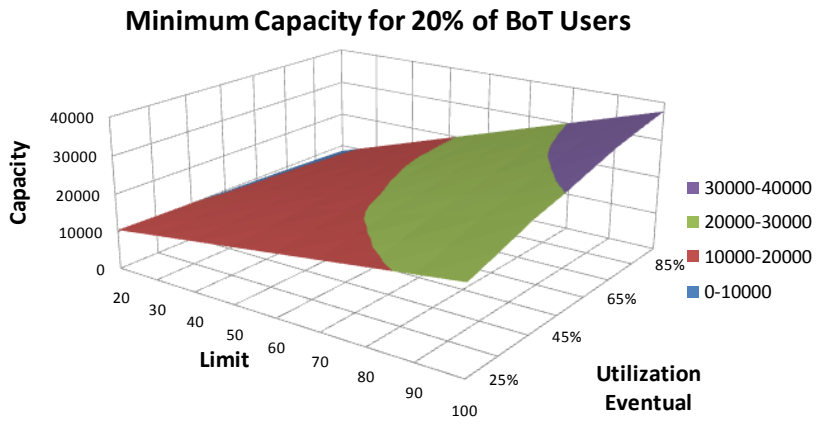
- *Hierarchical Generative Model*

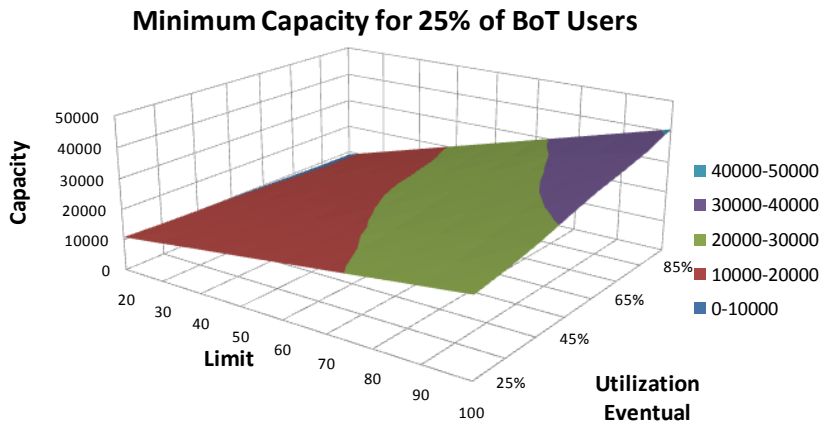
Utilization profiles of the synthetic workload population

Experiment #1: results



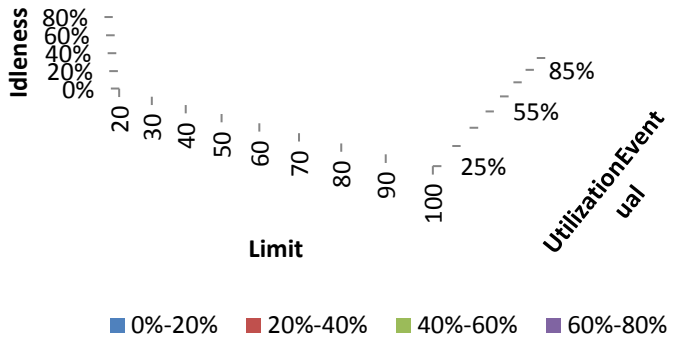




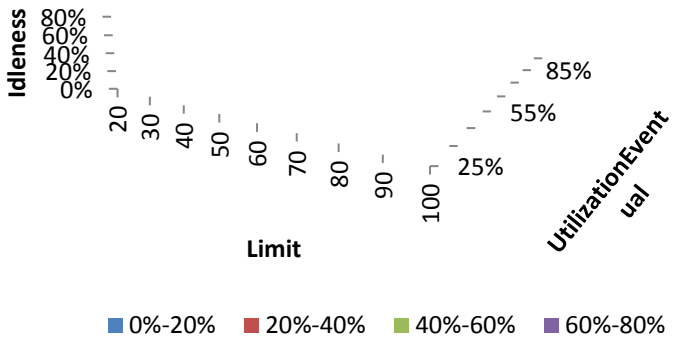


Experiment #2: results

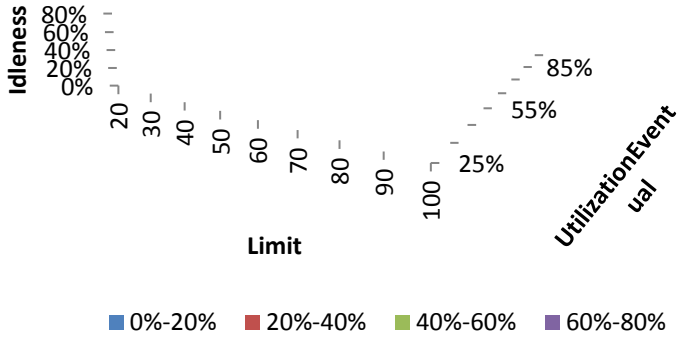
Idleness with 10% of BoT Users



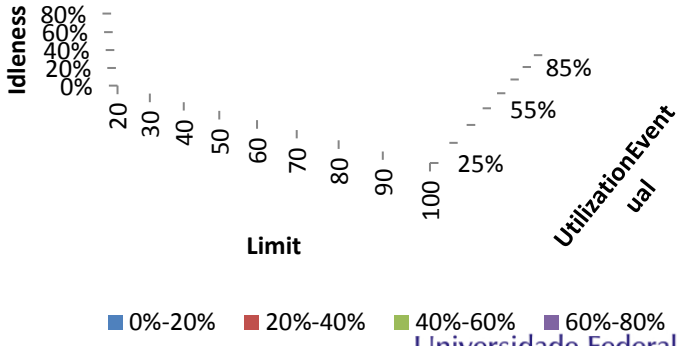
Idleness with 15% of BoT Users



Idleness with 20% of BoT Users

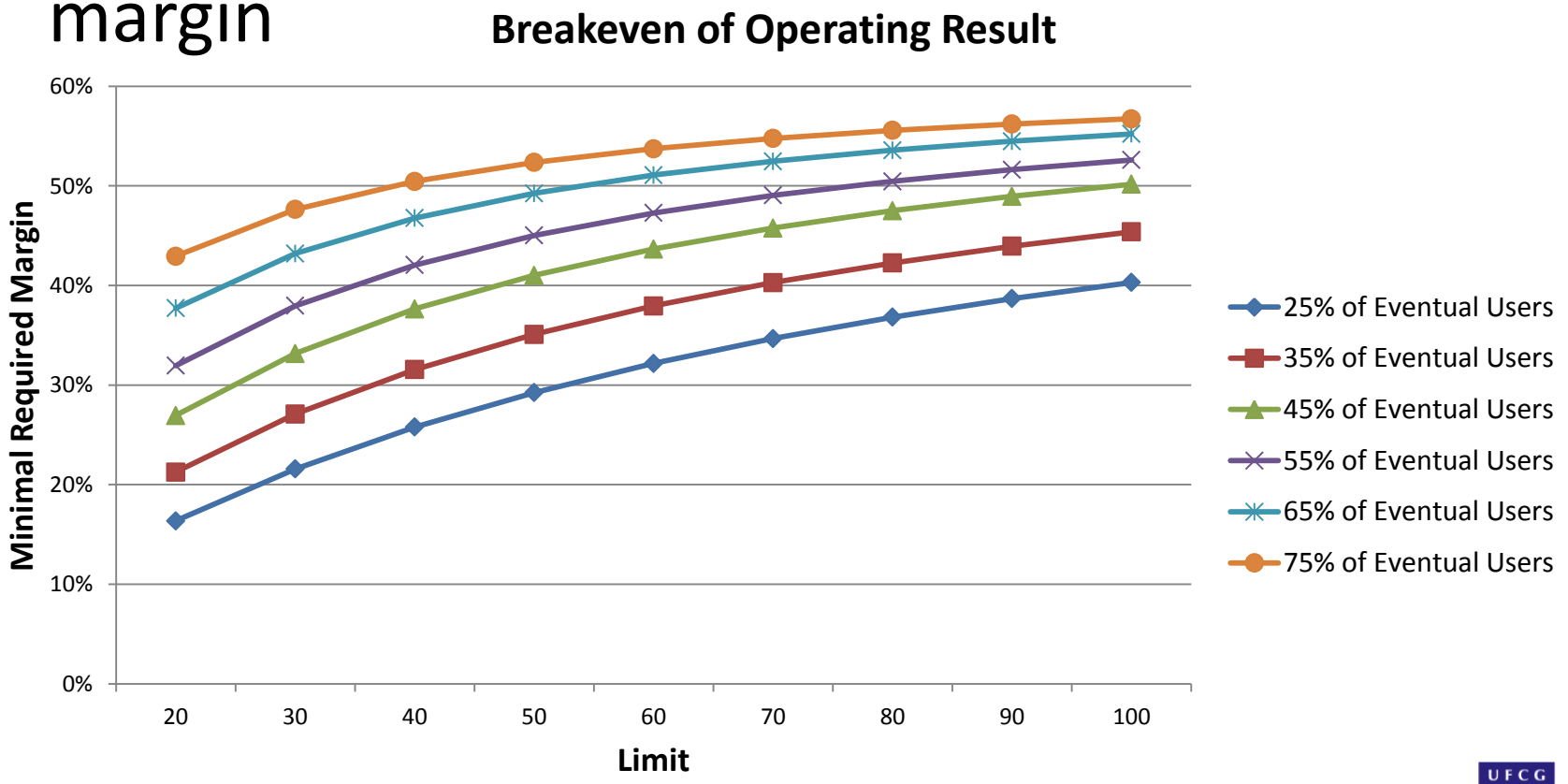


Idleness with 25% of BoT Users



Experiment #3: results

- Impact of the limit on the minimal required margin



Partial conclusions

- The limit value has a relevant impact on the cost of running the infrastructure
  - Availability cost
- Users with very intensive and eventual use (BoT) press the minimum capacity and increase the system idleness
- Maintained the same profile of the population and the limit value, the system dynamics is not dependent on the number of users
  - Scaling does not eliminate the problem

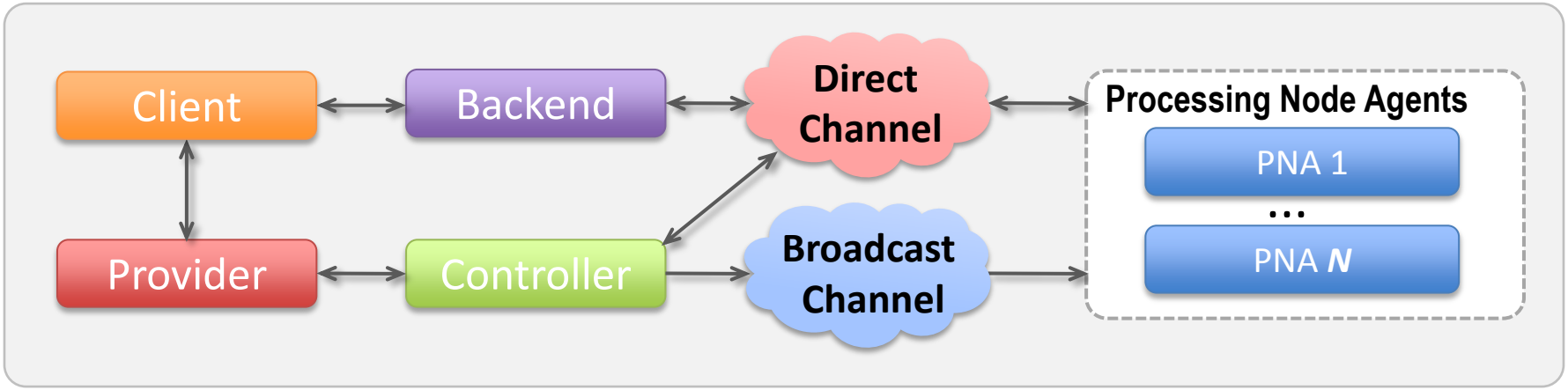
How can we appropriately serve BoT users in an IaaS setting?

- We have been investigating alternative approaches to build highly elastic IaaS providers
  - Federate resources owned by third parties that are not able to provide themselves an IaaS infrastructure over the eventual spare capacity existent
    - Availability cost may be irrelevant because it has already been amortized by the main business supported by the resources
      - Small datacenters, P2P grids, unconventional devices (STBs in DTV systems, smartphones, etc.)
- Cloud infrastructure is assembled "Just in Time" by a broker

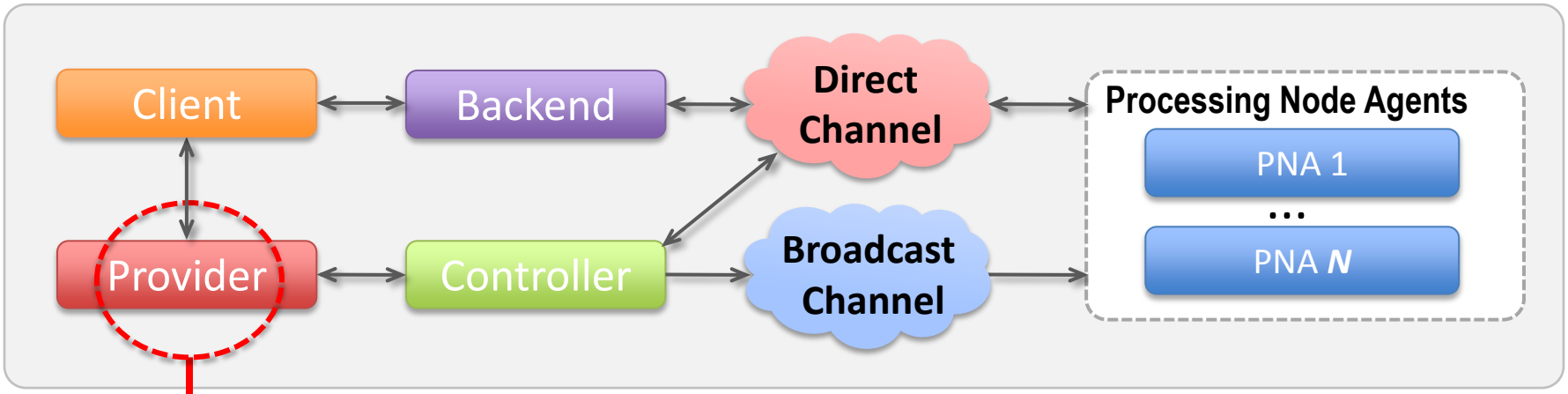
On-Demand Distributed Computing Infrastructure

- OddCI considers 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

OddCI Architecture

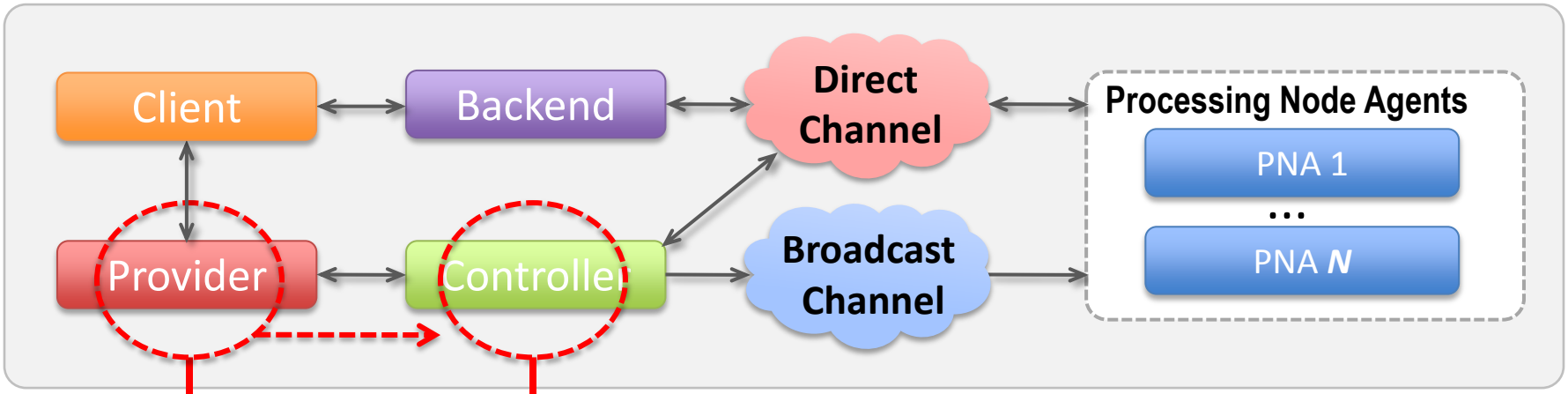


OddCI Architecture: operation



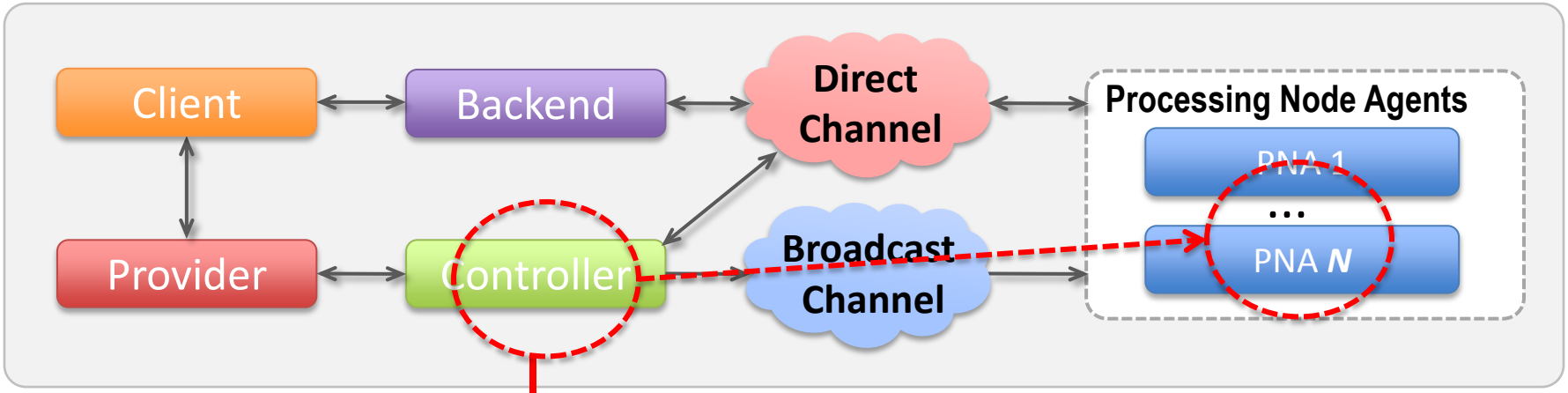
- Client submits a "processing request" to the **provider**
  - DCI instance size (number of processing nodes)
  - Application image, common data
  - Node requirements

OddCI Architecture: operation



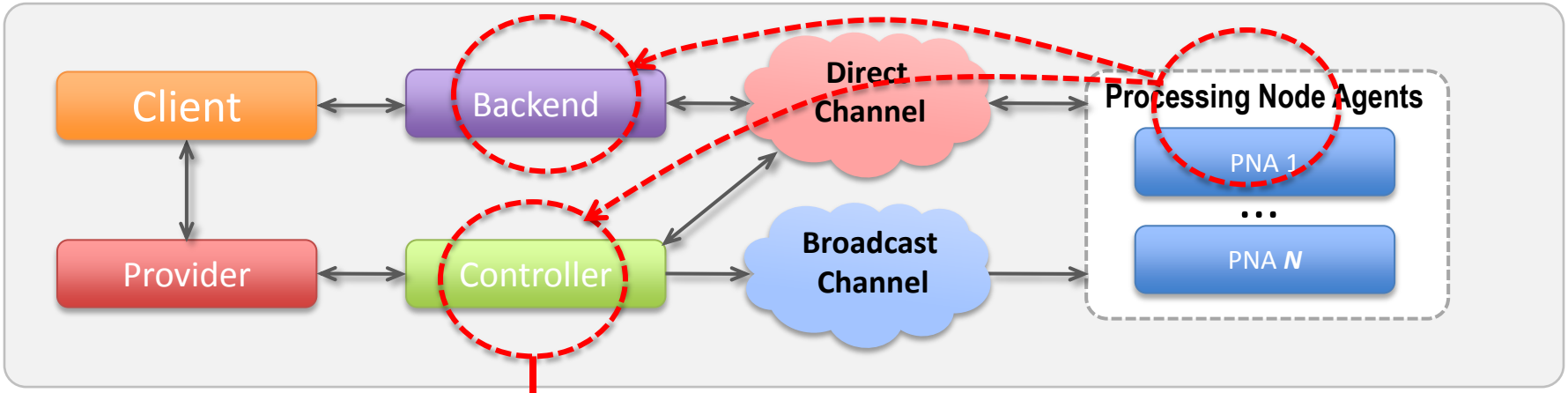
- **Provider** evaluates the client's request
  - authenticates the client
  - checks availability
  - keeps control information
  - commands the **Controller** for creating the required OddCI instance

OddCI Architecture: operation



- **Controller** triggers a wakeup message to ***PNAs*** through the broadcast channel
  - Wakeup message carry the PNA software and user application
- All **PNA** receive messages virtually at the same time
- **Controller** also sends other control messages (e.g. dismantle instances, leave, …)

OddCI Architecture: operation

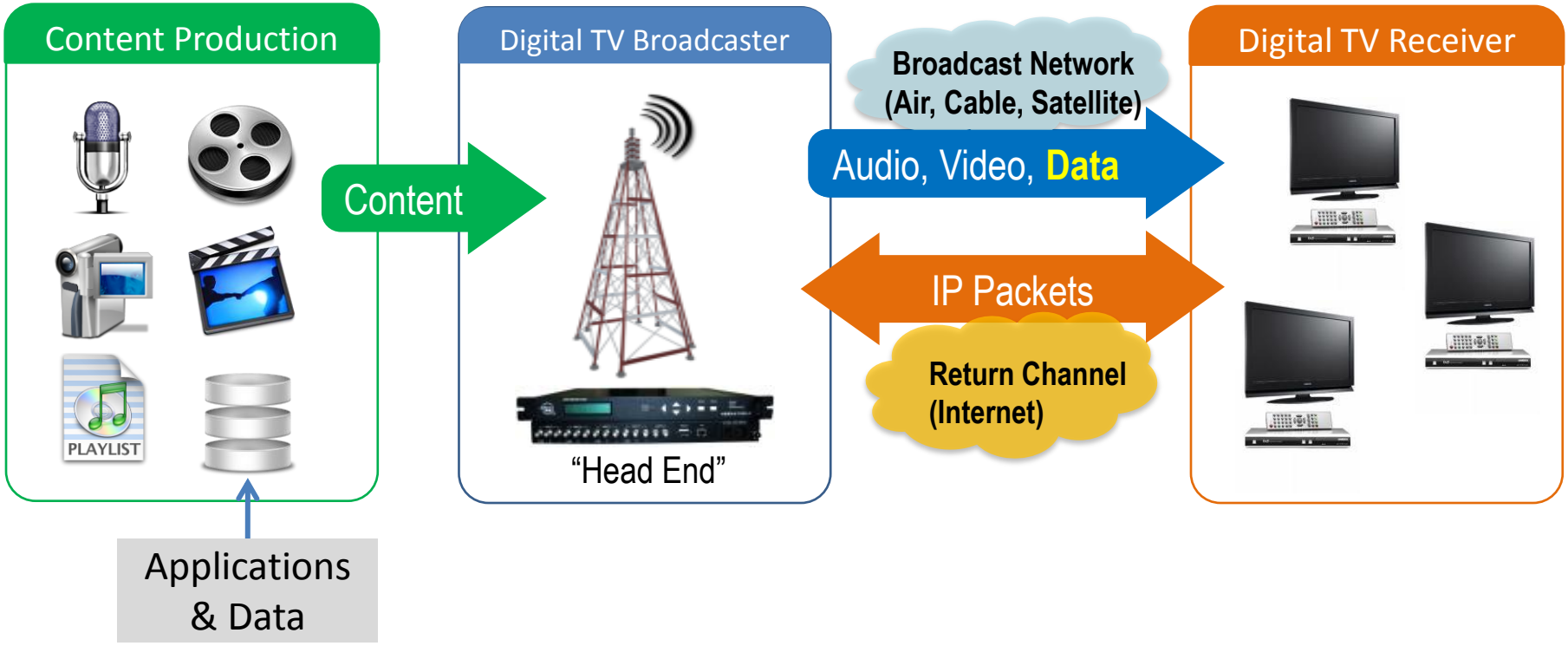


- **PNA** loads application image for execution in a DVE (Dynamic Virtual Environment)
- **Controller** monitors active **PNA**
- Use of the **Direct channel**
  - **Application** can interact with the **Backend** for requesting specific input data or send results
  - **PNA** sends status messages frequently to the **Controller**

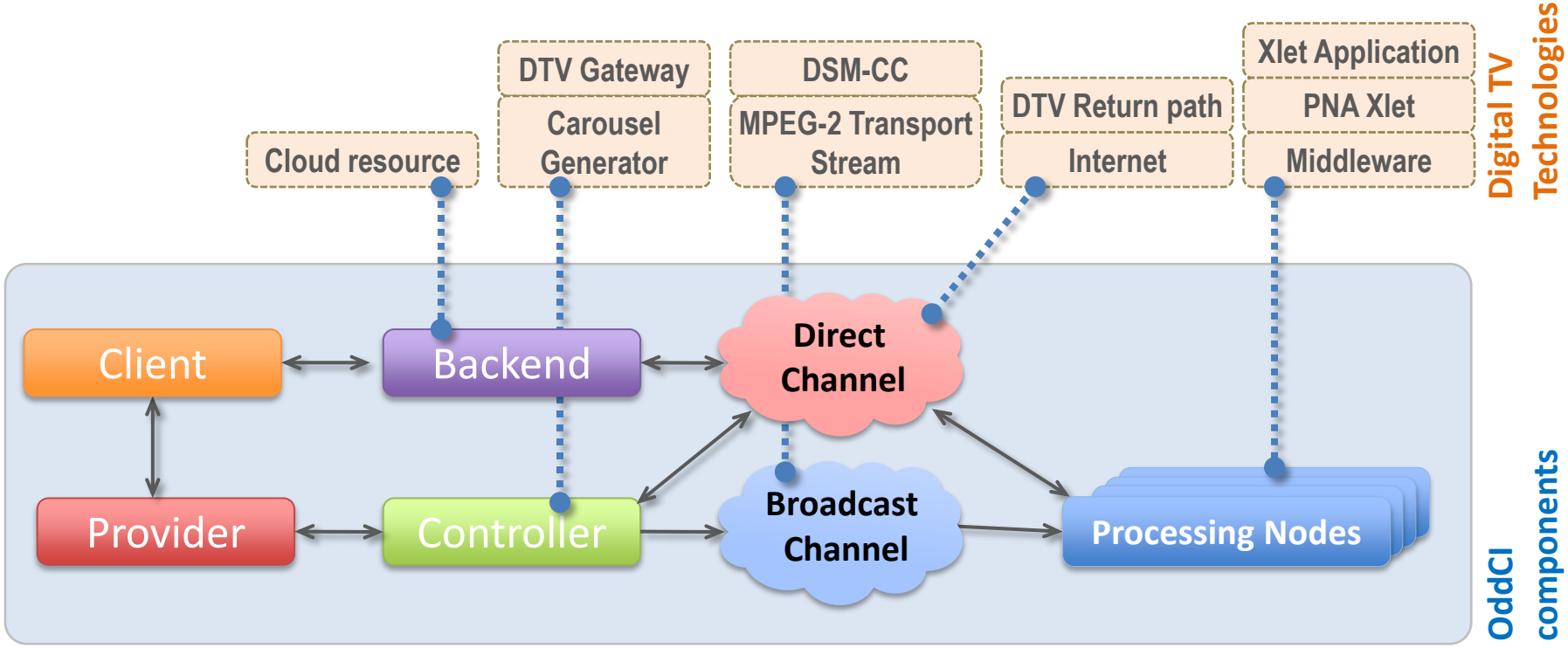
OddCI-Ginga: OddCI over a Digital TV Network

- Why DTV network?
  - Open technology, well-defined standards
  - Native transmission of data in broadcast
  - Fast expansion, being deployed in many countries
  - Potential for millions of devices
  - Powerful middleware
- And also ...
  - Feasibility for building a testbed
  - Previous experience of our group helping to develop Brazilian DTV middleware (SBTVD/Ginga)

DTV Generic Model



Implementing OddCI over DTV components



Performance assessment: testbed

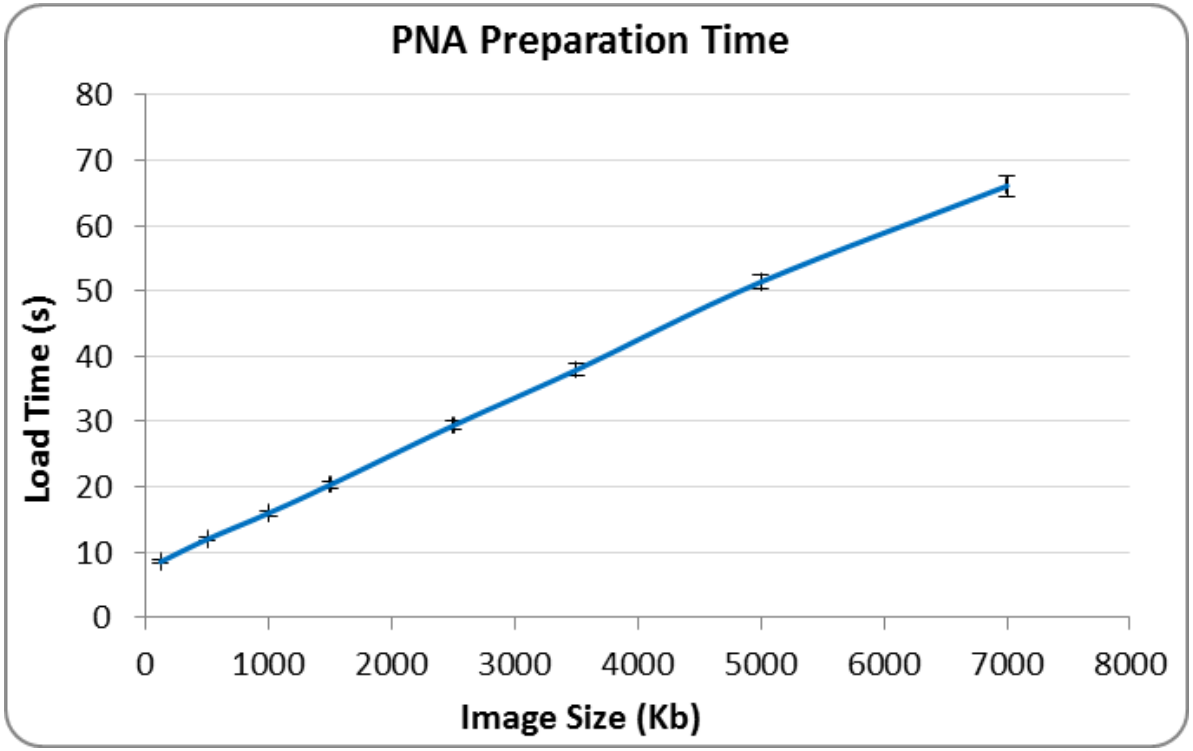
- TV Station
  - ISDB-T Digital Modulator (Brand/model: Linear ISMOD - Series ISCHIO)
  - Carousel Generator and Multiplexer (Brand/model: Domm Xstream)
  - Maximum data rate of data carousel set to 1Mbps
- Digital TV Receivers
  - Low-end: Proview XPS-1000, STi7001 processor 266MHz, 256MB RAM, Ethernet, STLinux
- Controller and Backend
  - Implemented as network services: Apache/Tomcat v6.0.33, HTTP for message exchange
  - Web framework Grails/Groovy scripts
  - MySQL v.5.1 for storing tasks and results in the backend.
- Provider
  - Web application for requesting the creation of instances and upload files to be sent to the PNA through the DTV data carousel

Performance assessment: Benchmarking

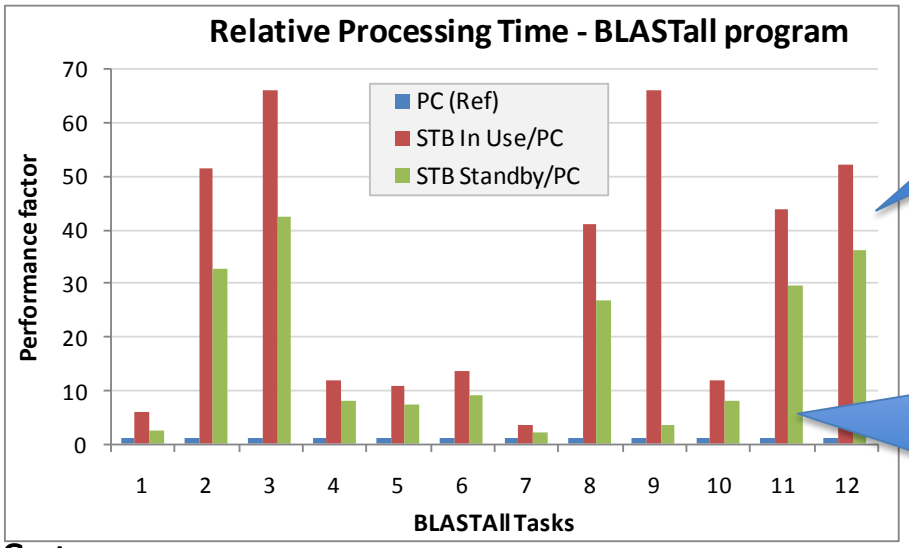
- Bioinformatics application
  - We ported the NCBI Toolkit (blastall and blastcl3 programs) to the low-end DTV receiver used
- Reference machine (for comparison with DTV receiver)
  - Notebook Intel (R) Core i3-2310M 2.1GHz, 4 GB RAM, Fast Ethernet, Ubuntu 11.10 64-bits
  - Same applications were tested and compared

Results: PNA load time (broadcast channel)

- Time for application load at PNA is linear with image (executable) size



Results: DTV receiver Performance



Lower is better

When in "stand by", DTV receiver is 65% faster than when "watching TV" mode

- Experiment Setup
  - Tests performed in a low-end DTV receiver (the cheapest in the Brazilian market)
- Remarks
  - Reference PC is ~23 times faster than DTV receiver in "in use" mode
  - Reference PC is ~12 times faster than DTV receiver in "standby" mode
  - Background applications does not interfere on the image/audio quality

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 individual 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)
- Many research challenges still opened

Current Work

- Small scale deploy at the city of João Pessoa
  - 100 STBs to test interactivity of the SBTVD
- Peer-to-peer JiT Clouds
  - Combine the ideas of peer-to-peer grids (such as OurGrid) with the federation of clouds
  - Provide the same user experience that users have when use spot instances in Amazon Web Services
    - Different business model (market vs resource exchange)

Thanks for your attention!

- Contact me at: fubica@dsc.ufcg.edu.br

Questions??