Cloud Computing: A Paradigm Shift

- Holistic system (social) view is passé
  - Tenants make resource acquisition decisions; no incentive to be fair or friendly to others
  - IaaS providers have no incentive to minimize tenant costs; they only react to market pressure
  - Challenge is to design the mechanisms that engender trust in the cloud marketplace

- Marketplace mechanisms
  - Fixed pricing for reserved resources
  - Fixed pricing for shared resources
  - Dynamic pricing for reserved resources

Fixed pricing for exclusive resources

"Pricing is per instance-hour consumed for each instance type. Partial instance-hours consumed are billed as full hours."

Marketplace Implications?

- 08:00 am / Amazon $3
- 09:00 am / Amazon $3
- 10:00 am / Amazon $2
- 11:00 am / Amazon $2

Hosts

Tasks
(Cloud) Colocation Games

- IaaS cloud providers offer fixed-sized instances for a fixed price
- Provider’s profit = number of instances sold; no incentive to colocate customers
- Virtualization enables colocation to reduce costs without QoS compromises
- Customers’ selfishness reduces the colocation process to a strategic game

Colocation Games: Questions

- Does it reach equilibrium?
- If so, how fast?
- If so, at what price (of anarchy)?
- How about multi-resource jobs/hosts?
- How about multi-job tasks?
- How about job/host dependencies?
- How could it be implemented?
- How would it perform in practice?
- ...

Colocation Game: Model

- A hosting graph \( G = (V, E) \)
  - \( V \) & \( E \) labeled by capacity vector \( R \) and fixed price \( P \)
- Workloads as task graphs \( T_i = (V_i, E_i) \)
  - \( V_i \) & \( E_i \) labeled by a utilization vector \( W \)
- Valid mappings
  - \( V_i \rightarrow V \) & \( E_i \rightarrow E; \Sigma W \leq R \); supply meets demand
- Shapley Cost function
  - Cost \( P \) of a resource is split among workloads mapped to it in proportion to use

\[
e_M(T_i) = \sum_{j \in (V_i, E_i)} P \frac{W_j}{V_j}
\]

The General Colocation Game (GCG)

- GCG is a pure strategies game:
  Each workload is able to make a (better response) “move” from a valid mapping \( M \) into another \( M' \) so as to minimize its own cost
- Example applications:
  - Overlay reservation, e.g., on PlanetLab
  - CDN colocation, e.g., on CloudFront
General Colocation Game: Properties

- GCG may not converge to a Nash equilibrium
- Theorem:
  Determining whether a GCG has a Nash Equilibrium is NP-Complete (by reduction to 3-SAT problem)
- Need more structure to ensure convergence

Colocation Games: Variants

- Process Colocation Game (PCG):
  Each workload consists of a single vertex (process) that needs to be assigned to a single host with only one capacitated resource
- Multidimensional PCG (MPCG):
  Same as PCG but with multi-capacitated resources
- Example applications:
  - VM colocation, e.g., on a Eucalyptus cluster
  - Streaming server colocation

Colocation Games: Variants

- Parallel PCG (PPCG):
  Task graph consists of a set of disconnected vertices (independent processes), each with multidimensional resource utilization needs
- Uniform PPCG:
  Same as PPCG but with identical resource utilization for all processes
- Example applications:
  - Map-Reduce paradigm
  - MPI scientific computing paradigm

Colocation Games: Theoretical results

- PCG converges to a Nash Equilibrium under better-response dynamics
- PCG converges to a Nash Equilibrium in $O(n^2)$ better-response moves, where $n = |V|$.
- Price of Anarchy for PCG is 3/2 when hosting graph is homogeneous and 2 otherwise
- MPCG converges to a Nash equilibrium under better-response dynamics
- Uniform PPCG converges to a Nash equilibrium under better-response dynamics
- ...
PCG: Better Response

Best-Response moves require knowledge of utilizations of all processes – not practical

Local Better-Response solution:
1. Select a random target hosting node and obtain process utilizations of all processes on that node
2. Determine if a cost-reducing “legal” move to that node is possible – an NP-hard Knapsack problem
   - Dynamic Programming solution in pseudo-polynomial time for small number (100s) of processes/host [DPKP]
   - Breadth-First branch & bound Search heuristic [BFS]
   - Depth-First branch & bound Search heuristic [DFS]

PCG: Performance Evaluation

Workloads
- Trace-driven: CoMon PlanetLab Traces
  - Real hosting environment with 3-dimensional resource utilizations
  - Infeasible to compute optimal colocation
- Synthetic: Perfectly Packable
  - Allows systematic exploration of the space
  - Optimal colocation is known by construction

Metrics (over 100 experiments)
- Colocation Ratio (bounded by PoA)
  - How inefficient is the resulting colocation compared to optimal or best?
- Number of moves (not migrations) until NE is reached
  - How much churn (overhead) to be expected?

PCG: Synthetic baseline results

<table>
<thead>
<tr>
<th>Number of processes</th>
<th>Median (Colocation Ratio)</th>
<th>Median (Number of Moves)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BFS</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>DFS</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>DP KP</td>
<td>2</td>
</tr>
</tbody>
</table>

MPCG: PlanetLab baseline results

<table>
<thead>
<tr>
<th>Number of processes</th>
<th>Median (Colocation Ratio)</th>
<th>Median (Number of Moves)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BFS</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>DFS</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>DP KP</td>
<td>2</td>
</tr>
</tbody>
</table>
The CLOUDCOMMONS prototype

- **API for Strategic Services**
  To facilitate colocation, e.g., allow users to find each other, compute strategic responses, ...

- **API for Operational Services**
  To enforce outcomes of colocation, e.g., migration, reconfiguration, accounting, ...

- **Implementation over Xen**
  Xen Colocation Services (XCS)

---

CLOUDCOMMONS: Benefit to Customers

- Planet-Lab trace-driven experiments (Overheads/costs of all XCS services included)

  - **At most 7% of customers overpay less than 1%**
  - **50% of customers save more than 68%**

---

How do we depart from prior work?

- **Cooperative cost-sharing games**
  - Find coalition where nobody gains by leaving
  - Computationally hard
  - Applied to best-effort routing problems
  - Player cost not use based; unjustifiable

---

Fixed pricing for shared resources

The perils of the fixed pricing model

- It’s here to stay; metered pricing rejected

- Implications:
  - Customer has no incentive to save bandwidth
  - ISP cost depends on peak demand – 95/5 rule
  - Reigning in bandwidth hogs is incompatible with Net Neutrality

- Must devise mechanisms that take ISPs out of the “traffic shaping” business

DSLAM “last-mile” architecture

Solution: Create a marketplace

- Recognize the two types of user utilities:
  - Interactive Traffic (IT)
    - Browsing, VoIP, Video, Messaging, Gaming, ...
    - Limited bandwidth; highly sensitive to response time
  - Fluid Traffic (FT)
    - P2P, Network backup, Netflix/software downloads, ...
    - Open-ended bandwidth; less sensitive to response time

- Create a marketplace:
  1. Give users rights to DSLAM bandwidth, and
  2. Let users trade IT & FT allocations over time
The Marketplace

- Each user gets a fixed budget per epoch
  - Budget ($B_i$) proportional to level of service
  - An epoch is a fixed number of time-slots, e.g., 1 day = 288 5-min slots

- Trade & Cap
  - User engages in a pure strategies game that yields a schedule for its IT sessions
  - User acquires as much FT bandwidth as its remaining budget would allow

Trading Phase: Strategy Space

- Session:
  - An IT session is the sequence of slots during which an IT application is active

- Slack:
  - User may have flexibility in scheduling IT sessions; slack specifies the number of slots that an IT session is allowed to be shifted back/forth

- Strategy Space:
  - The set of all possible arrangements of IT sessions within allowable slack define the strategy space for a user

Trading Phase: Cost Function

- Let $x_{jk}$ be the bandwidth used in slot $k$ by a chosen IT session schedule for user $i$.
- The cost incurred by user $i$ is given by:

\[ c_i = \sum_{k \in \text{slots}} x_{jk} \cdot U_k = \sum_{k \in \text{slots}} x_{jk} \left( \sum_{j \in \text{users}} x_{jk} \right) \]

- Cost of user $i$ depends on the choices made by other users – hence the game!

Trading Phase: Illustration

Cost(User 2) = 2*2 + 1*2 = 6
Trading Phase: Illustration

Cost(User 2) = 1*2 + 1*1 + 1*1 = 4

Trading Phase: Best Response

- BR of user $i$ is the schedule of IT sessions that minimizes its cost $c_i$
- Computing BR is NP-hard, equivalent to solving a generalized knapsack problem
- Dynamic programming solution is pseudo-polynomial in the product of the number of sessions and number of slots
- Scales well for all practical settings – 100s of users and 100s of slots

Trading Phase: Findings

- Provably converges to Nash Equilibrium, even in presence of constraints
- For $n$ users, Price of Anarchy is $n$, but in practice below 2, especially for $n > 10$
- Experimentally, large reduction of peak utilization, even with small flexibility

Capping Phase: Best Response

- BR of user $i$ is to maximize total FT allocation

$$ w_i = \sum_{k \in \text{slots}} w_{ik} $$

subject to the budget constraint

$$ \sum_{k \in \text{slots}} w_{ik} \cdot \left( U_0 + \sum_{j \in \text{users}} w_{jk} \right) = B_i - c_i $$
Capping Phase: Findings

- Computing BR is efficient using Lagrange Multipliers method
- Provably, converges to a unique global (social) optimum that maximizes the FT allocations of all users
- Experimentally, smoothes the aggregate IT+FT traffic to any desirable level controlled by a “resistance parameter” R

Trade & Cap: Implementation

- On Client Side (e.g., DSL Modem):
  - Strategic agent to execute Trade & Cap
  - Operational service to profile, classify, and shape

- ISP Side (e.g., DSLAM or BRAS):
  - Support exchange between strategic agents
  - Enforce total traffic/slot/user from Trade & Cap

User Input:
- As simple as checking box to join marketplace, and as elaborate as micromanaging RT slacks
- May set a fraction of “budget” as insurance

Client-side Profiler:
- May be explicitly controlled by applications (or user settings)

Client-side Traffic Shaper:
- Work-conserving (not reservation based) Linux Hierarchical Token Bucket (HTB)
- Allows FT to use underutilized RT bandwidth
Experimental Evaluation

Workload
Derived from WAN traces of MAWI project

- Identify users from volume and direction of flows to known ports (e.g., most traffic destined to port 80)
- Identify user IT sessions using thresholds on per-IP traffic intensities over time
- Slack introduced using various models (e.g., fixed, proportional, etc.)

Trading Phase: Experimental PoA

Theoretical PoA is \( n \) but not in practice

Trading Phase: Smoothing effect

Value proposition to ISPs

<table>
<thead>
<tr>
<th>Max Slack</th>
<th>Reduction in 95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>15%</td>
</tr>
<tr>
<td>6</td>
<td>24%</td>
</tr>
<tr>
<td>12</td>
<td>31%</td>
</tr>
</tbody>
</table>

Trade & Cap: Flexibility pays off!

Value proposition to customers
Trade & Cap: Beyond DSLAMs

- **Trade & Cap is a general mechanism**
  It can be used to coordinate how a shared resource is used by selfish parties who are not subject to the “pay as you go” model – e.g., “fixed pricing”

- **Examples**
  - Coordinating consumption of “reserved” versus “fluid” (CPU/network) capacities of VMs sharing a single host
  - Coordinating “reserved” versus “fluid” bandwidth utilization by multiple ISP customers (e.g., enterprises)

Fixed Pricing: Alternatives?

- **Vickrey-style auctions work†**
  - Assumes supply < demand
  - Takes a social perspective
  - Offers a strategy-proof solution
  - Requires central authority
  - Susceptible to collusion


Dynamic pricing: Better mechanism?

- **Customer cost should be a function of supply and demand**
  - Supply may vary over time
  - Supplier’s cost may vary over time
  - Demand may vary over time
  - Demand may exhibit structure, and may be subject to malleable constraints

- **Need language to specify supply and demand (and act as basis for SLAs)**

Resource Supply/Demand Model

- **Supply/demand SLA types: (C, T, D, W)**
  - C ~ amount available or consumed
  - T ~ allocation period
  - D ~ tolerable number of missed allocations in W
  - W ~ window of >= 1 allocation intervals

- **Examples**
  - SLA type (2,5,0,1)
    2 resource units supplied/consumed every 5 seconds with no missed allocations allowed
  - SLA type (3,30,2,5)
    3 resource units supplied/consumed every 30 seconds with no more than 2 out of 5 missed allocations
SLA Calculus

- Models various patterns of allocation and consumption (e.g., RR, GPS, LB, …)

- SLA types define type hierarchies
  - $(1, N, 0, 1) < (k, k + N, 0, 1)$
  - $(C, T, D, W) < (C, T, D', W'$) if $D < D'$
  - ...

- Possible to transform SLAs from one form to another (safer) form

Using SLA Calculus for Colocation

- Not possible to colocate
  - \[
  \begin{array}{|c|c|c|c|c|c|}
  \hline
  & Job 1 & Job 2 & Job 3 & Job 4 & Job 5 \\
  \hline
  C & 1 & 2 & 3 & 4 & 5 \\
  T & 4 & 9 & 17 & 34 & 67 \\
  \hline
  \end{array}
  \]

- Possible to colocate
  - \[
  \begin{array}{|c|c|c|c|c|c|}
  \hline
  & Job 1 & Job 2 & Job 3 & Job 4 & Job 5 \\
  \hline
  C & 1 & 2 & 3 & 4 & 5 \\
  T & 4 & 8 & 16 & 32 & 64 \\
  \hline
  \end{array}
  \]

- SLA types and calculus provide a notion of supply & demand elasticity

Morphing SLAs for Efficiency

MorphoSys: Performance

- Allow Relocation
- Morph Co-Tenants
- Morph Once @ Arrival
Beyond Simple Types

- A workload is a set of requests (tasks), each with its SLA, subject to constraints:
  - Temporal dependencies between tasks
  - Start and end times

- Flexibilities might exist; another source of elasticity:
  - Min and max delays between tasks
  - Deadline slacks

The Customer's Perspective

- Why should customers expose the elasticity of their workloads?
- Current IaaS (fixed) pricing mechanisms do not provide proper incentives

- Implications:
  - Less efficient workload management
  - Customers (should) game the marketplace

Dynamic Pricing: Shapley Value

- Well defined concept for fair cost sharing from coalitional game theory
  - Marginal contribution to the total cost, averaged over every permutation, e.g., for 3 workloads

\[
s(w_i) = \frac{1}{6} \left( 2c(w_i) + [c(w_2w_1) - c(w_2)] + [c(w_3w_1) - c(w_2)] + [c(w_2w_3) - c(w_2)] + [c(w_2w_3) - c(w_2)] + [c(w_2w_3) - c(w_2)] \right)
\]

- Impractical to calculate
- Estimate by sampling random permutations
Workload Elasticity = Savings

<table>
<thead>
<tr>
<th>Workload Type</th>
<th>Shapley</th>
<th>Utopian</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cost Incurred (for uniform mix of workloads)

5/28/2014

CLOUDCOMMONS: http://csr.bu.edu/cc

Supported by a NSF Large CyberTrust Award (in collaboration with Brown and UCI)

Problem with today’s cloud solutions

Public clouds are “closed” & prescriptive

- Stock hardware (hard to justify GPUs or a BlueGene)
- Special computational models (MapReduce, or?)
- Operational data is not visible (limits innovation)
- Irrational economic models (why fixed pricing?)
- Designed to lock in customers
- Perform poorly on HPC and big data
- Not conducive to federation of data assets
- Security by obscurity

Not conducive to experimentation/innovation
Can we change this status quo?

Need an ecosystem for

- **Infrastructure Providers**
  EMC, IBM, Dell, NTAP, Cisco, Intel, AMD, …

- **Software Platforms**
  VMware, RedHat, Hadoop, Azure, MapReduce…

- **Aggregators and Resellers**
  XCS, Morphosys, CloudPack, …

- **Customers**
  HPC apps, Web apps, Enterprise, Gov, …

Open Cloud: A Paradigm Shift

- **Multiple implementation and operational models**
  - Competing platforms suited to HPC, big data, …
  - Open economic model to create an efficient market
  - Operational data visible to various stakeholders

- **Ecosystem needs a facility for “mashups”**

Open Cloud eXchange: Architecture

Massachusetts Green HPC Facility

Operational since: February 2013
The Massachusetts Open Cloud

An initiative of The BU Hariri Institute for Computing in collaboration with MGHPCC
- Project is spearheaded by Center for Cloud Innovations (Orran Krieger, Director)
- Project leverages three Institute research priorities: Cybersecurity, Big Data, and Cloud Computing

A great vehicle for inter/national collaboration
- Federation interest from Oakridge, Brazil, ...

Conclusion

- Resource management must be seen in an economics context
- By setting up the right mechanisms, one can engender trust in the cloud marketplace
- Supply elasticity meets demand elasticity for an efficient marketplace
- New services needed to support strategic and operational aspects of new mechanisms
- Need to experiment at scale!

Partial Paper Trail...


http://www.bu.edu/hic