CloudShare: Guaranteed Application Performance on Idle Data Center Resources

D. Anderson[⋆], D. Kondo[†], A. Legrand[†]

⋆ SSL, UC Berkeley † CNRS - Inria - Grenoble

Berkeley – Inria – Stanford Workshop 2013

Outline

Outline

Why would Business need Distributed Computing Systems?

Originally, little need for performance

- Business computations seldom extend beyond ordinary rational arithmetic (unless when science is involved in business) Mostly desktop usage
- Computer systems distributed iff the company is (interconnect business units)

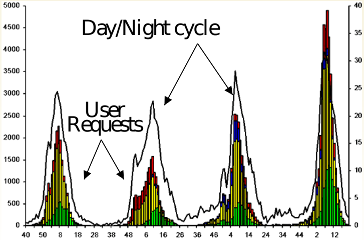
Why would Business need Distributed Computing Systems?

Originally, little need for performance

- Business computations seldom extend beyond ordinary rational arithmetic (unless when science is involved in business) Mostly desktop usage
- Computer systems distributed iff the company is (interconnect business units)

And then came the Internet

- Some company relying on the Internet emerged (eBay, amazon, google)
- Computers naturally play a central role in their business plan
- Cannot afford to loose clients $\rightsquigarrow$ High Availability Computing



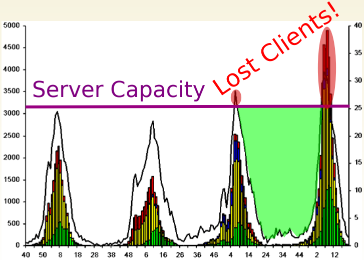
Why would Business need Distributed Computing Systems?

Originally, little need for performance

- Business computations seldom extend beyond ordinary rational arithmetic (unless when science is involved in business) Mostly desktop usage
- Computer systems distributed iff the company is (interconnect business units)

And then came the Internet

- Some company relying on the Internet emerged (eBay, amazon, google)
- Computers naturally play a central role in their business plan
- Cannot afford to loose clients ⇝ High Availability Computing
- But load is very changing ⇝ Servers dimensioned for flash crowds

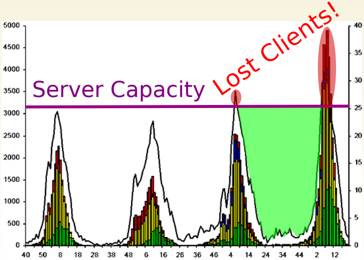


Originally, little need for performance

- Business computations seldom extend beyond ordinary rational arithmetic (unless when science is involved in business) Mostly desktop usage
- Computer systems distributed iff the company is (interconnect business units)

And then came the Internet

- Some company relying on the Internet emerged (eBay, amazon, google)
- Computers naturally play a central role in their business plan
- Cannot afford to loose clients ⇝ High Availability Computing
- But load is very changing ⇝ Servers dimensioned for flash crowds



Amazon idea

- Data centers often 85% idle
- Rent unused power to others!
- Computers better amortized. Buy bigger ones, loose no client
- Infrastructure as a Service (IaaS)

Here Comes the Cloud

Client Incitatives

- Complexity of infrastructure management hidden from users
  IT maintenance burden assumed by external specialists
- Pay only used power: rent a server 1h, send computations in the cloud, enjoy

This is called Elastic Computing

- The created need revealed very profound: everyone wants it now
- Clients even want to rent OS+apps (PaaS) or software (SaaS)

Here Comes the Cloud

Client Incitatives

- Complexity of infrastructure management hidden from users
  IT maintenance burden assumed by external specialists
- Pay only used power: rent a server 1h, send computations in the cloud, enjoy

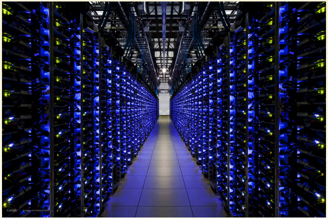
This is called Elastic Computing

- The created need revealed very profound: everyone wants it now
- Clients even want to rent OS+apps (PaaS) or software (SaaS)

The Data Centers Growth

- Scale allows Cost Cuttings, as always. (Motivation for big DC already existed)
- Clouds removes the wastes due to over-dimensioning

  $\Rightarrow$ Corporate Data Centers become as big as Scientific Supercomputers!



Google Data Center

Here Comes the Cloud

Client Incitatives

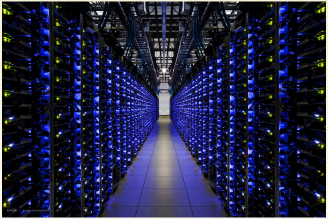
- Complexity of infrastructure management hidden from users
  IT maintenance burden assumed by external specialists
- Pay only used power: rent a server 1h, send computations in the cloud, enjoy

This is called Elastic Computing

- The created need revealed very profound: everyone wants it now
- Clients even want to rent OS+apps (PaaS) or software (SaaS)

The Data Centers Growth

- Scale allows Cost Cuttings, as always. (Motivation for big DC already existed)
- Clouds removes the wastes due to over-dimensioning
  ⇒ Corporate Data Centers become as big as Scientific Supercomputers!



Google Data Center

It's not that sunny Cloud infrastructures are not that easy and transparent to use (virtualization and co-localization overhead, unexpected preemption of spot instances, unavailability) and can quickly reveal expensive

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

Volunteer computing: harnessing the free collective computational and storage resources of desktop PC's throughout the Internet

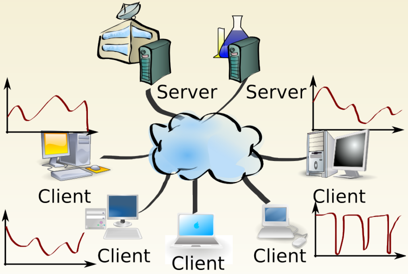
- Cooperation $\rightsquigarrow$ one of the largest and most powerful distributed computing systems on the planet

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

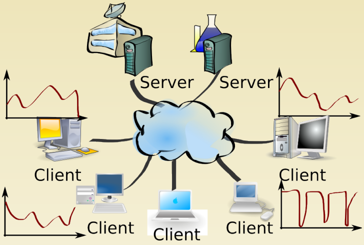
Volunteer computing: harnessing the free collective computational and storage resources of desktop PC's throughout the Internet

- Cooperation $\rightsquigarrow$ one of the largest and most powerful distributed computing systems on the planet



- Volunteer donate their unused CPU cycles to scientific/geeky/humanitarian projects
- Complex client and server scheduling mechanisms to handle practical considerations (e.g., heterogeneity, volatility, volunteer satisfaction).
- Understanding the behavior of such architectures is non-trivial

BOINC: the most popular VC infrastructure

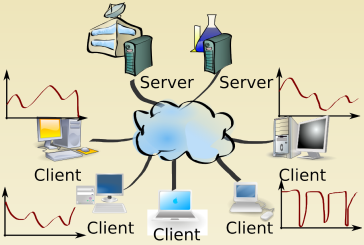


The Berkeley Open Infrastructure for Network Computing is the *most popular* VC infrastructure today:

- 50 projects: SETI@home, WCG, Einstein@home, ClimatePrediction.net, ...
- $596,000$ hosts, 9.2 PetaFlops (March 2013)
- Since 2000, generated 100$+$ scientific publications (Science, Nature)

BOINC has proved to scale to **millions** of unreliable resources

BOINC: the most popular VC infrastructure



The Berkeley Open Infrastructure for Network Computing is the *most popular* VC infrastructure today:

- 50 projects: SETI@home, WCG, Einstein@home, ClimatePrediction.net, ...
- $596,000$ hosts, 9.2 PetaFlops (March 2013)
- Since 2000, generated 100+ scientific publications (Science, Nature)

BOINC has proved to scale to **millions** of unreliable resources

VC limitations Unfortunately, the types of applications and services that can run over VC platforms is largely limited to trivially parallel ones

Fair and autonomous scheduling of billions of CPU-bound independent tasks (i.e. optimize **throughput**)

Extending to a wider context requires smart modeling and scheduling techniques

Associated Team Backgrounds

The cloud and VC context have actually a lot in common but proposing good solutions requires the good blend of practice and theory.

Berkeley/Palo Alto Lead development of BOINC middleware for volunteer computing. Google data-center management. Invualuable knowledge of production systems.

- David Anderson, Walfredo Cirne

Associated Team Backgrounds

The cloud and VC context have actually a lot in common but proposing good solutions requires the good blend of practice and theory.

Berkeley/Palo Alto Lead development of BOINC middleware for volunteer computing. Google data-center management. Invualuable knowledge of production systems.

- David Anderson, Walfredo Cirne

Inria MESCAL Statistical performance and failure modeling, simulation, scheduling, game theory and resource management.

- Derrick Kondo, Arnaud Legrand, Bruno Gaujal, Jean-Marc Vincent, Olivier Richard
- Sheng Di, Bahman Javadi, B. Donnassolo, Lucas Schnorr

Associated Team Backgrounds

The cloud and VC context have actually a lot in common but proposing good solutions requires the good blend of practice and theory.

Berkeley/Palo Alto Lead development of BOINC middleware for volunteer computing. Google data-center management. Invualuable knowledge of production systems.

- David Anderson, Walfredo Cirne

Inria MESCAL Statistical performance and failure modeling, simulation, scheduling, game theory and resource management.

- Derrick Kondo, Arnaud Legrand, Bruno Gaujal, Jean-Marc Vincent, Olivier Richard
- Sheng Di, Bahman Javadi, B. Donnassolo, Lucas Schnorr

Inria associated team (2009-2014)

CloudComputing@Home Create a virtually dedicated cloud from unreliable Internet resources

CloudShare Guaranteed Application Performance on Idle Data Center Resources

Expected outcomes

Models and Algorithms

- Models of bursty workloads and resource usage
- Statistical and machine learning algorithms for predicting idleness in data centers
- Fair scheduling algorithms for guaranteed performance across unreliable resources

Traces and Software Tools

- Failure and Application Trace Archive
- Cloud and VC Simulator
- BOINC software adapted to data centers

In the following, I will present a joint work (CCGrid'11) with B. Donnassolo and C. Geyer, from UFRGS, Porto Alegre, Brazil.

Outline

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

Volunteer computing: harnessing the free collective computational and storage resources of desktop PC's throughout the Internet

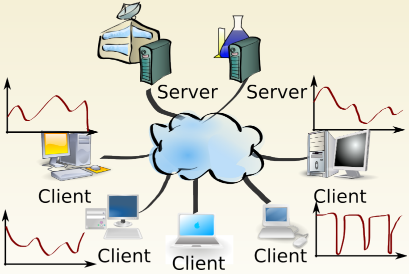
- Cooperation $\rightsquigarrow$ one of the largest and most powerful distributed computing systems on the planet

Volunteer Computing in a Nutshell

Most of the world's computing power is distributed across the *hundreds of millions* of Internet hosts on *residential broadband* networks

Volunteer computing: harnessing the free collective computational and storage resources of desktop PC's throughout the Internet

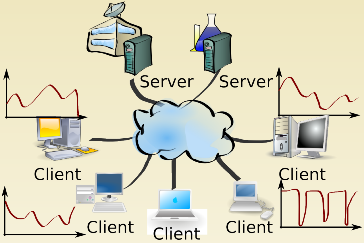
- Cooperation ⇝ one of the largest and most powerful distributed computing systems on the planet



- Volunteer donate their unused CPU cycles to scientific/geeky/humanitarian projects
- Complex client and server scheduling mechanisms to handle practical considerations (e.g., heterogeneity, volatility, volunteer satisfaction).

BOINC: the most popular VC infrastructure

The Berkeley Open Infrastructure for Network Computing is the *most popular* VC infrastructure today:

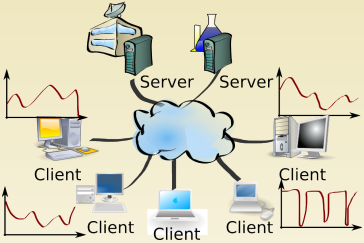


- 50 projects: SETI@home, WCG, Einstein@home, ClimatePrediction.net, ...
- $596,000$ hosts, 9.2 PetaFlops (March 2013)
- Since 2000, generated 100+ scientific publications (Science, Nature)

BOINC has proved to scale to **millions** of unreliable resources

BOINC: the most popular VC infrastructure

The Berkeley Open Infrastructure for Network Computing is the *most popular* VC infrastructure today:



- 50 projects: SETI@home, WCG, Einstein@home, ClimatePrediction.net, ...
- 596,000 hosts, 9.2 PetaFlops (March 2013)
- Since 2000, generated 100+ scientific publications (Science, Nature)

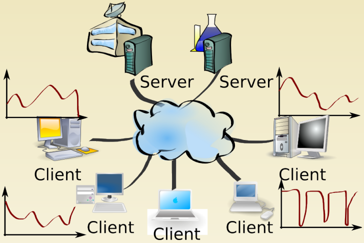
BOINC has proved to scale to **millions** of unreliable resources

Unfortunately, the types of applications and services that can run over VC platforms is largely limited to trivially parallel ones

Fair and autonomous scheduling of billions of CPU-bound independent tasks (i.e. optimize **throughput**)

BOINC: the most popular VC infrastructure

The Berkeley Open Infrastructure for Network Computing is the *most popular* VC infrastructure today:



- 50 projects: SETI@home, WCG, Einstein@home, ClimatePrediction.net, ...
- $596,000$ hosts, 9.2 PetaFlops (March 2013)
- Since 2000, generated 100+ scientific publications (Science, Nature)

BOINC has proved to scale to **millions** of unreliable resources

Unfortunately, the types of applications and services that can run over VC platforms is largely limited to trivially parallel ones

Fair and autonomous scheduling of billions of CPU-bound independent tasks (i.e. optimize **throughput**)

Possible research direction: Fair optimization of **response time** of BoTs

The BOINC Server in a Nutshell

Each project sets up its own server

The BOINC Server in a Nutshell

Each project sets up its own server, which uses pragmatic scheduling mechanism to handle:

Volatility and Heterogeneity

- The server waits for clients to contact him
- Upon work request, the server selects a subset of tasks and assign them a soft deadline
- If a task is not returned before its deadline, it is considered as lost and may be resubmitted to another client

The BOINC Server in a Nutshell

Each project sets up its own server, which uses pragmatic scheduling mechanism to handle:

Volatility and Heterogeneity

- The server waits for clients to contact him
- Upon work request, the server selects a subset of tasks and assign them a soft deadline
- If a task is not returned before its deadline, it is considered as lost and may be resubmitted to another client

Rewarding Volunteers

- Clients claim credits based on benchmark
- Servers reward minimum of claimed credits for correct results
- Rank volunteers based on their contribution

The BOINC Server in a Nutshell

Each project sets up its own server, which uses pragmatic scheduling mechanism to handle:

Volatility and Heterogeneity

- The server waits for clients to contact him
- Upon work request, the server selects a subset of tasks and assign them a soft deadline
- If a task is not returned before its deadline, it is considered as lost and may be resubmitted to another client

Rewarding Volunteers

- Clients claim credits based on benchmark
- Servers reward minimum of claimed credits for correct results
- Rank volunteers based on their contribution

Correctness (over-clocking, unstable numerical applications, malicious participants)

- Majority voting
- Limited replication, homogeneous redundancy, black hole

The BOINC Client in a Nutshell

Each volunteer may register to many projects and define resource shares

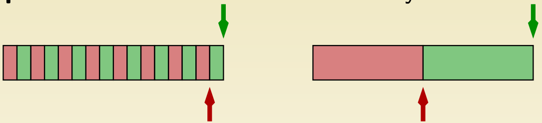
The BOINC Client in a Nutshell

Each volunteer may register to many projects and define resource shares

Fairness **Respect resource shares** and variety $\leadsto$ Fair Sharing...

Each volunteer may register to many projects and define resource shares

Fairness **Respect resource shares** and variety $\rightsquigarrow$ Fair Sharing...



The BOINC Client in a Nutshell

Each volunteer may register to many projects and define resource shares

Fairness **Respect resource shares** and variety $\rightsquigarrow$ Fair Sharing...

Satisfy deadlines Rough simulation and switch to *Earliest Deadline First* if needed

The BOINC Client in a Nutshell

Each volunteer may register to many projects and define resource shares

Fairness **Respect resource shares** and variety $\rightsquigarrow$ Fair Sharing...

Satisfy deadlines Rough simulation and switch to *Earliest Deadline First* if needed

Avoid waste Work as much as possible and do not start working on tasks whose deadline can obviously not be met

The BOINC Client in a Nutshell

Each volunteer may register to many projects and define resource shares

Fairness **Respect resource shares** and variety $\rightsquigarrow$ Fair Sharing...

Satisfy deadlines Rough simulation and switch to *Earliest Deadline First* if needed

Avoid waste Work as much as possible and do not start working on tasks whose deadline can obviously not be met

Consequence

Once a task has been downloaded, the client will try to complete it before its deadline

A project with shorter deadline could thus obtain more resources than the volunteer wishes

Long term fairness inhibits requesting tasks to overworked projects

The Deadline Effect

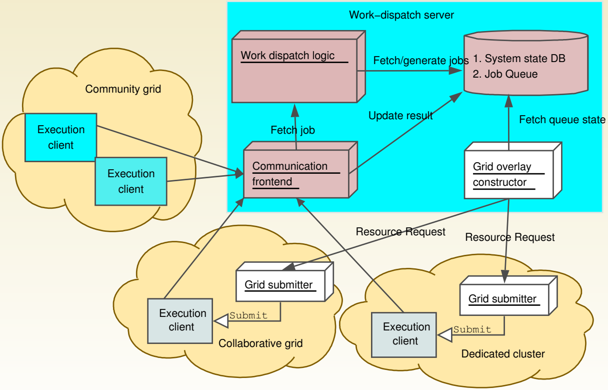
- The slack is the ratio between the deadline and the actual running time of tasks [KAM07] (has to be > 7; the current median is about 60)

BOINC is perfectly tailored for throughput optimization

But with such a slack, response time is really large

GridBot [SSGS09] (Technion - Israel Institute of Technology)

- Focus on response time of BoTs
- Use both community resources (BOINC) and grid resources (Condor). Has also been connected with Amazon EC2
- Better than BOINC and than Condor for this kind of workload

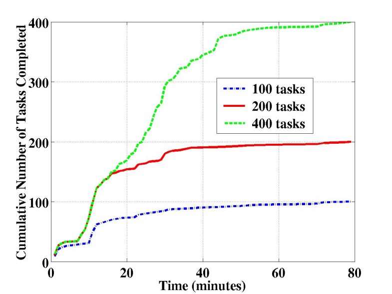


GridBot

GridBot [SSGS09] (Technion - Israel Institute of Technology)

- Focus on response time of BoTs
- Use both community resources (BOINC) and grid resources (Condor). Has also been connected with Amazon EC2
- Better than BOINC and than Condor for this kind of workload

FCFS scheduling on a desktop Grid [KTB+04]

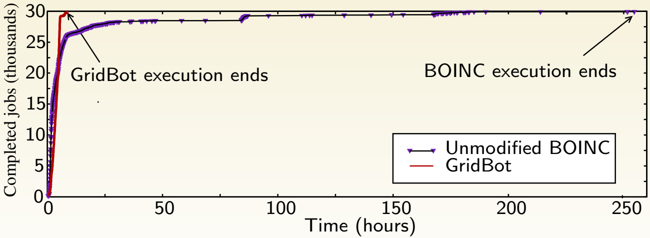


A.k.a the last finishing task issue

GridBot

GridBot [SSGS09] (Technion - Israel Institute of Technology)

- Focus on response time of BoTs
- Use both community resources (BOINC) and grid resources (Condor). Has also been connected with Amazon EC2
- Better than BOINC and than Condor for this kind of workload
- Tighter deadlines for reliable resources
- Replicate on reliable resources *toward the end*



GridBot

GridBot [SSGS09] (Technion - Israel Institute of Technology)

- Focus on response time of BoTs
- Use both community resources (BOINC) and grid resources (Condor). Has also been connected with Amazon EC2
- Better than BOINC and than Condor for this kind of workload
- Tighter deadlines for reliable resources
- Replicate on reliable resources *toward the end*

The deadline/boomerang effect

"Since a single client is connected to many such projects, those with shorter deadlines (less than three days) effectively require their jobs to be executed immediately, thus postponing the jobs of the other projects. This is considered selfish and leads to contributor migration and a bad project reputation, which together result in a significant decrease in throughput."

GridBot

GridBot [SSGS09] (Technion - Israel Institute of Technology)

- Focus on response time of BoTs
- Use both community resources (BOINC) and grid resources (Condor). Has also been connected with Amazon EC2
- Better than BOINC and than Condor for this kind of workload
- Tighter deadlines for reliable resources
- Replicate on reliable resources *toward the end*

Some (guru) volunteers noticed that tight deadline jobs were causing significant delays in other projects and even deadline misses

- Are current mechanisms sufficient to isolate projects from each others ?
- Response-time optimizing strategies (deadline, replication) need to be accepted by volunteers and other projects

A Game Theoretic Model of BOINC

Although every client tries to fairly and efficiently share its resources, the configuration decisions of each project may impact the performance of other projects

A Non Cooperative Game This can be modeled as a game between the projects

- Each project should choose its **own** scheduling strategy (deadline, replication, resource selection, . . .) to optimize its **own** metric
- This is a long term game
- The volunteer opinion and feeling really matters

A Game Theoretic Model of BOINC

Although every client tries to fairly and efficiently share its resources, the configuration decisions of each project may impact the performance of other projects

A Non Cooperative Game This can be modeled as a game between the projects

- Each project should choose its **own** scheduling strategy (deadline, replication, resource selection, . . .) to optimize its **own** metric
- This is a long term game
- The volunteer opinion and feeling really matters

Methodology

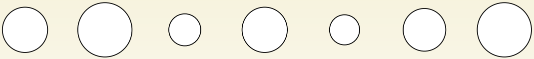
- Really hard to study by deploying a real system
- Really hard to study on a purely theoretical point of view

We used SimGrid, a simplified but realistic modeling of BOINC, real traces from the FTA, and realistic application characteristics

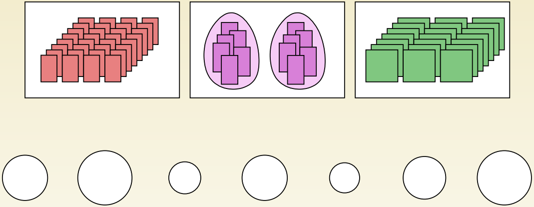
Modeling the Whole System

Volunteer V_j:

- **peak performance** (in $MFLOP.s^{-1}$)
- an **availability** trace



Modeling the Whole System



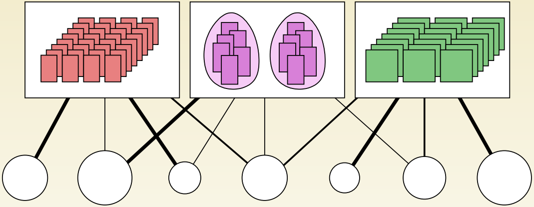
Volunteer V_j:

- **peak performance** (in $MFLOP.s^{-1}$)
- an **availability** trace

Project P_i:

- Obj_i: **objective function**: either **throughput** ϱ_i or the **average completion time of a** $batch$ α_i
- w_i $[MFLOP.task^{-1}]$: **size of a task**
- b_i $[task.batch^{-1}]$: **number of tasks within each batch**
- r_i $[batch.day^{-1}]$: **input rate**, i.e., the number of batches per day

Modeling the Whole System

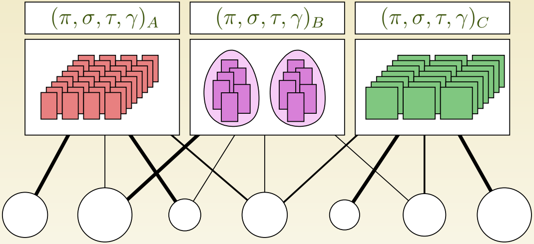


Volunteer V_j:

- **peak performance** (in $MFLOP.s^{-1}$)
- an **availability** trace
- **project shares**

Project P_i:

- Obj_i: **objective function**: either **throughput** ϱ_i or the **average completion time of a** *batch* α_i
- w_i $[MFLOP.task^{-1}]$: **size of a task**
- b_i $[task.batch^{-1}]$: **number of tasks within each batch**
- r_i $[batch.day^{-1}]$: **input rate**, i.e., the number of batches per day

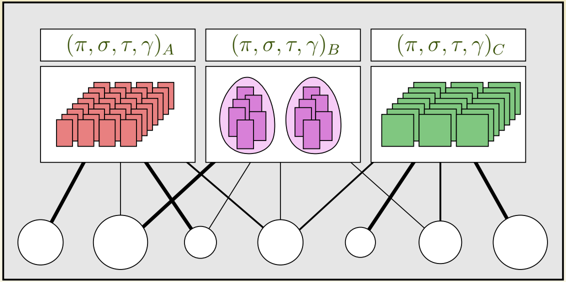


Strategy S_i

- π_i: **work send policy** [KAM07] ($\pi_{cste=c}$/saturation/EDF)
- σ_i: **slack** [KAM07] ($s \in [1, 10]$)
- τ_i: **conn. interval** [HAH09] (12mn to 30hrs)
- γ_i: **replication strategy** [KCC07] ($r \in \{1, \ldots, 8\}$)

The **strategy** S_i of a project P_i is thus a tuple $(\pi, \sigma, \tau, \gamma)$

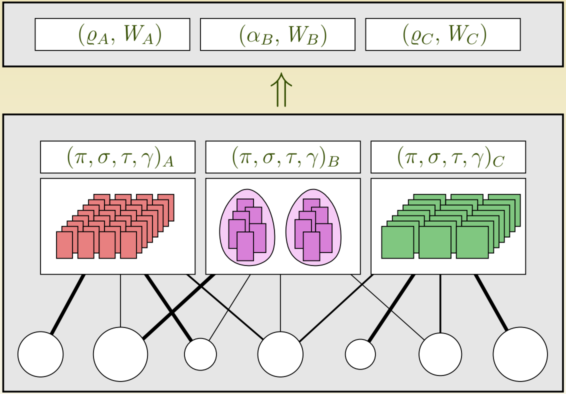
Modeling the Whole System



Strategy S_i

- π_i: **work send policy** [KAM07] ($\pi_{cste=c}$/saturation/EDF)
- σ_i: **slack** [KAM07] ($s \in [1, 10]$)
- τ_i: **conn. interval** [HAH09] (12mn to 30hrs)
- γ_i: **replication strategy** [KCC07] ($r \in \{1, \ldots, 8\}$)

The **strategy** S_i of a project P_i is thus a tuple $(\pi, \sigma, \tau, \gamma)$



Strategy S_i

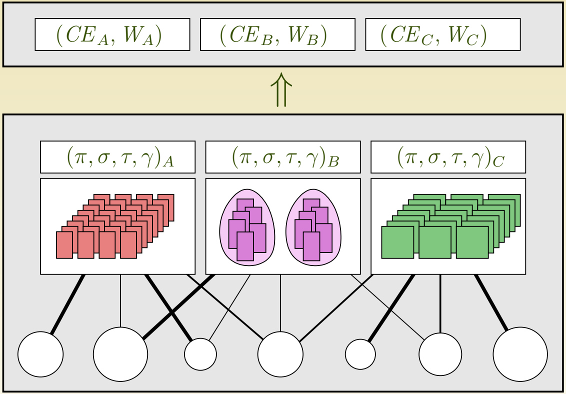
- π_i: **work send policy** [KAM07] ($\pi_{cste=c}$/saturation/EDF)
- σ_i: **slack** [KAM07] ($s \in [1, 10]$)
- τ_i: **conn. interval** [HAH09] (12mn to 30hrs)
- γ_i: **replication strategy** [KCC07] ($r \in \{1, \dots, 8\}$)

The **strategy** S_i of a project P_i is thus a tuple $(\pi, \sigma, \tau, \gamma)$

Outcome

- **Waste**
- Throughput/Response Time

Modeling the Whole System



Strategy S_i

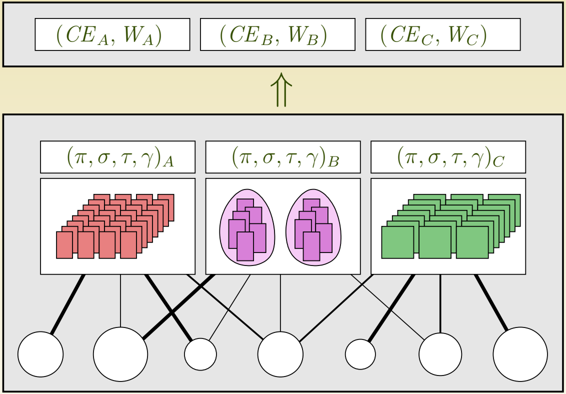
- π_i: **work send policy** [KAM07] ($\pi_{cste=c}$/saturation/EDF)
- σ_i: **slack** [KAM07] ($s \in [1, 10]$)
- τ_i: **conn. interval** [HAH09] (12mn to 30hrs)
- γ_i: **replication strategy** [KCC07] ($r \in \{1, \ldots, 8\}$)

The **strategy** S_i of a project P_i is thus a tuple $(\pi, \sigma, \tau, \gamma)$

Outcome

- **Waste**
- Throughput/Response Time $\rightsquigarrow$ **Cluster Equivalence** [KTB+04]

Modeling the Whole System



A complex problem

A multi-parametric, multi-player, multi-objective setting

Strategy S_i

- π_i: **work send policy** [KAM07] ($\pi_{cste=c}$/saturation/EDF)
- σ_i: **slack** [KAM07] ($s \in [1, 10]$)
- τ_i: **conn. interval** [HAH09] (12mn to 30hrs)
- γ_i: **replication strategy** [KCC07] ($r \in \{1, \ldots, 8\}$)

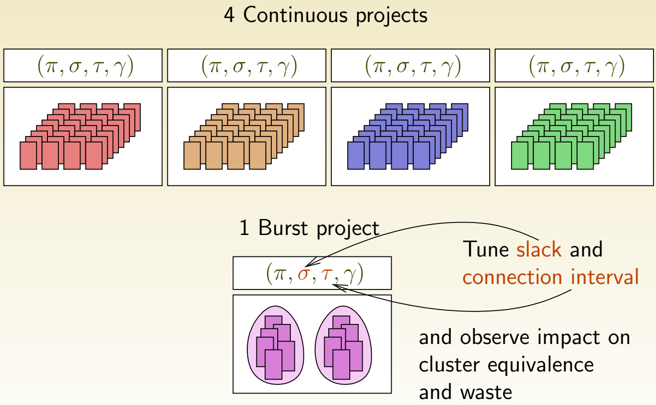
The **strategy** S_i of a project P_i is thus a tuple $(\pi, \sigma, \tau, \gamma)$

Outcome

- **Waste**
- Throughput/Response Time $\rightsquigarrow$ **Cluster Equivalence** [KTB+04]

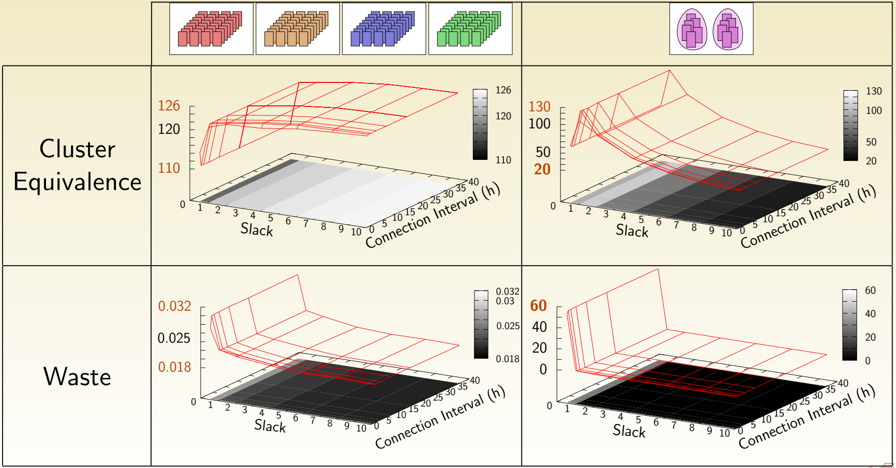
Sensibility Analysis

- 1000 clients over 5 months
- 4 identical throughput projects with standard configuration
- **1 burst project adjusting its slack and connection interval parameters** (fixed send policy and no replication)



Sensibility Analysis

- 1000 clients over 5 months
- 4 identical throughput projects with standard configuration
- **1 burst project adjusting its slack and connection interval parameters** (fixed send policy and no replication)



Sensibility Analysis

- 1000 clients over 5 months
- 4 identical throughput projects with standard configuration
- **1 burst project adjusting its slack and connection interval parameters** (fixed send policy and no replication)

Similar studies at finer granularity and for other parameters enable to understand that:

- σ: Slack has a dramatic effect on CE of all projects but a reasonable trade-off can be found (around 1.1)

Burst projects need to carefully tune their slack

- τ: Connection interval has almost no influence and can be arbitrarily set to 1hr
- γ: Allowing a few replicas (around 2-3) improves CE and waste
- π: Among the different work send policies we tried, one of them leads to unacceptably high waste (around 50%) for a minor CE improvement and should thus be disregarded as it wastes resources and could upset volunteers.

Best Response Strategy and Nash Equilibrium

Definition: Nash Equilibrium.

S is a **Nash equilibrium** for (V, P) iff

$$\text{for all } i \text{ and for any } S'_i,\ CE_i(V, P, S_{|S_i=S'_i}) \leqslant CE_i(V, P, S_i),$$

where $S_{|S_i=S'_i}$ denote the strategy set where P_i uses strategy S'_i and every other player keeps the same strategy as in S.

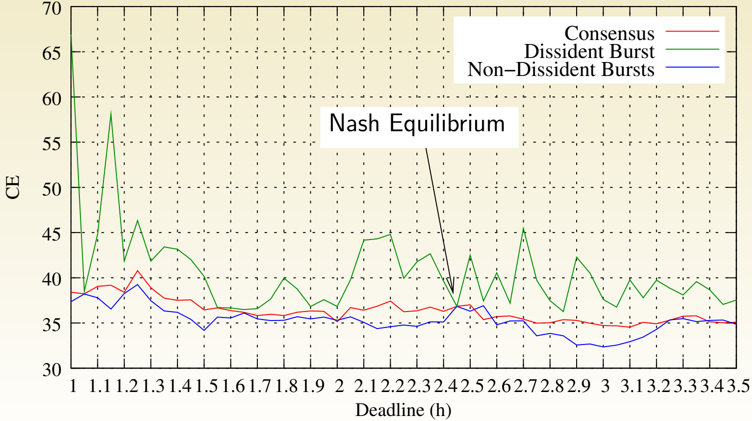
- a Nash equilibrium is a stable point for a **best response strategy**
- a **best response strategy** does not necessarily converge
- there may be no Nash equilibrium
- Nash equilibria are in the general case **neither fair nor efficient**
- Although they are not particularly desirable, they are adapted to model our situation

Utility Set Sampling and Nash Equilibrium

- 2 identical throughput projects with standard configuration
- **4 identical burst project adjusting their slack** (EDF send policy, replication=2)
- **Almost saturated system**

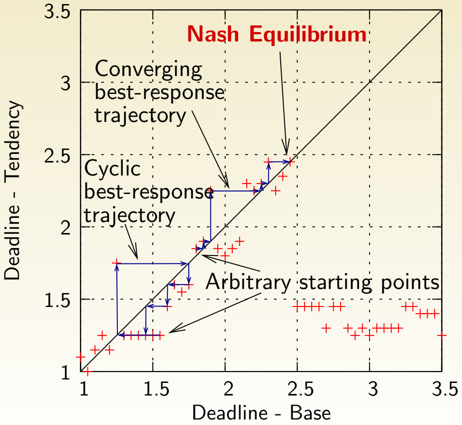
Utility Set Sampling and Nash Equilibrium

- 2 identical throughput projects with standard configuration
- **4 identical burst project adjusting their slack** (EDF send policy, replication=2)
- **Almost saturated system**



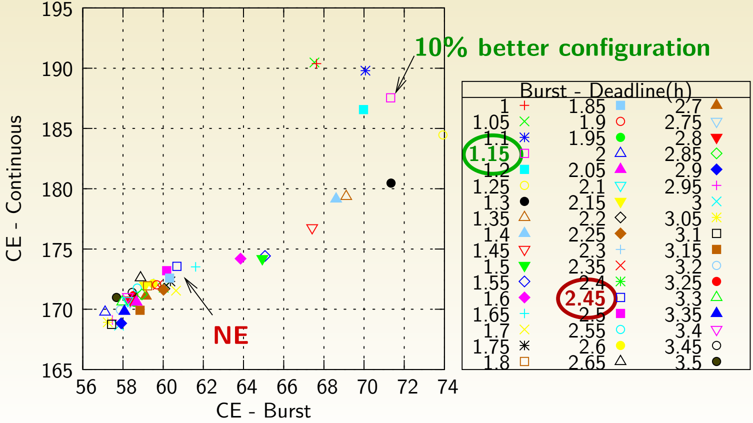
Utility Set Sampling and Nash Equilibrium

- 2 identical throughput projects with standard configuration
- **4 identical burst project adjusting their slack** (EDF send policy, replication=2)
- **Almost saturated system**



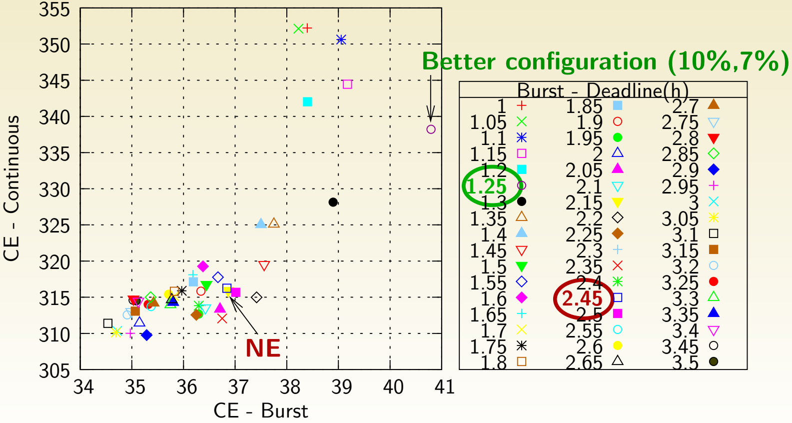
Utility Set Sampling and Nash Equilibrium

- 2 identical throughput projects with standard configuration
- **4 identical burst project adjusting their slack** (EDF send policy, replication=2)
- **Almost saturated system**



Utility Set Sampling and Nash Equilibrium

- 1 identical throughput projects with standard configuration
- **7 identical burst project adjusting their slack** (EDF send policy, replication=2)
- **Almost saturated system**



Conclusion

Harmful non-cooperative optimization Under high load and high pressure from burst projects, the current BOINC scheduling mechanism is unable to enforce fairness and project isolation We found inefficient Nash Equilibrium:

- Efficient configurations seem rather unstable.
- Can we found worse than 10% inefficiency?
- Could there be Braess paradoxes?

Game theory provides nice tools to address such issues
(correlated equilibria, pricing mechanisms, coalition)

Conclusion

Harmful non-cooperative optimization Under high load and high pressure from burst projects, the current BOINC scheduling mechanism is unable to enforce fairness and project isolation We found inefficient Nash Equilibrium:

- Efficient configurations seem rather unstable.
- Can we found worse than 10% inefficiency?
- Could there be Braess paradoxes?

Game theory provides nice tools to address such issues (correlated equilibria, pricing mechanisms, coalition)

Fair Optimization of Bag of Tasks A profound need: umbrella projects. Need to leverage both volunteer, grid and cloud resources. Currently designing fair multi-user scheduler for this context.

Conclusion

Harmful non-cooperative optimization Under high load and high pressure from burst projects, the current BOINC scheduling mechanism is unable to enforce fairness and project isolation We found inefficient Nash Equilibrium:

- Efficient configurations seem rather unstable.
- Can we found worse than 10% inefficiency?
- Could there be Braess paradoxes?

Game theory provides nice tools to address such issues (correlated equilibria, pricing mechanisms, coalition)

Fair Optimization of Bag of Tasks A profound need: umbrella projects. Need to leverage both volunteer, grid and cloud resources. Currently designing fair multi-user scheduler for this context.

Evolution of the Associated Team

- Collaboration with Google on cloud load characterization is on hold (Derrick Kondo is on sabbatical in the Bay area)
- Upcoming collaboration between B. Gaujal (Inria), R. Righter (UCB) and D. Anderson on reliable data storage in BOINC

Conclusion

Harmful non-cooperative optimization Under high load and high pressure from burst projects, the current BOINC scheduling mechanism is unable to enforce fairness and project isolation We found inefficient Nash Equilibrium:

- Efficient configurations seem rather unstable.
- Can we found worse than 10% inefficiency?
- Could there be Braess paradoxes?

Game theory provides nice tools to address such issues (correlated equilibria, pricing mechanisms, coalition)

Fair Optimization of Bag of Tasks A profound need: umbrella projects. Need to leverage both volunteer, grid and cloud resources. Currently designing fair multi-user scheduler for this context.

Evolution of the Associated Team

- Collaboration with Google on cloud load characterization is on hold (Derrick Kondo is on sabbatical in the Bay area)
- Upcoming collaboration between B. Gaujal (Inria), R. Righter (UCB) and D. Anderson on reliable data storage in BOINC

References

Eric Heien, David Anderson, and Kenichi Hagihara.
Computing low latency batches with unreliable workers in volunteer computing environments.
Journal of Grid Computing, 7:501–518, 2009.

Derrick Kondo, David P. Anderson, and John VII McLeod.
Performance evaluation of scheduling policies for volunteer computing.
In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing e-Science'07*, Bangalore, India, dec 2007.

Derrrick Kondo, Andrew A. Chien, and Henri Casanova.
Scheduling task parallel applications for rapid application turnaround on enterprise desktop grids.
Journal of Grid Computing, 5(4), 2007.

Derrrick Kondo, M. Taufer, C. Brooks, Henri Casanova, and Andrew A. Chien.
Characterizing and Evaluating Desktop Grids: An Empirical Study.
In *Proc. of the Intl. Parallel and Distributed Processing Symp. (IPDPS)*, 2004.

Mark Silberstein, Artyom Sharov, Dan Geiger, and Assaf Schuster.
Gridbot: execution of bags of tasks in multiple grids.
In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, New York, NY, USA, 2009. ACM.