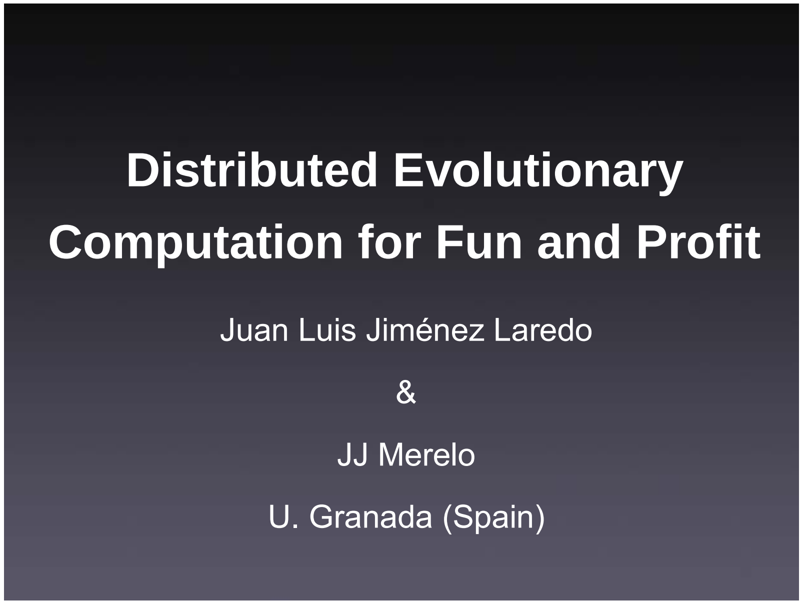


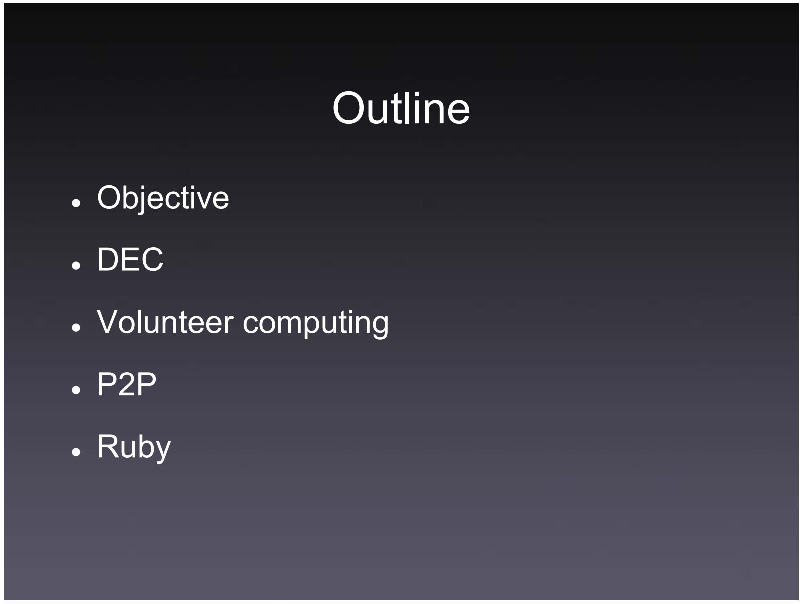
Distributed Evolutionary Computation for Fun and Profit

Juan Luis Jiménez Laredo

&

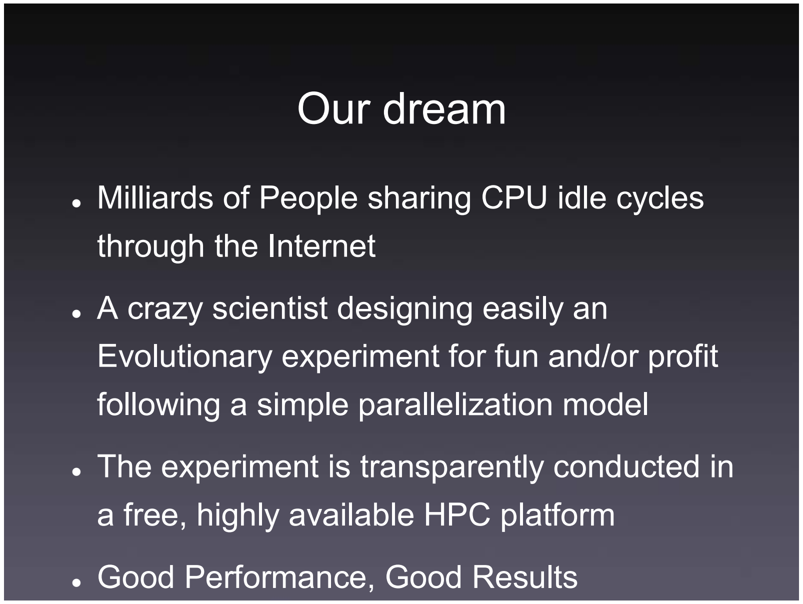
JJ Merelo

U. Granada (Spain)



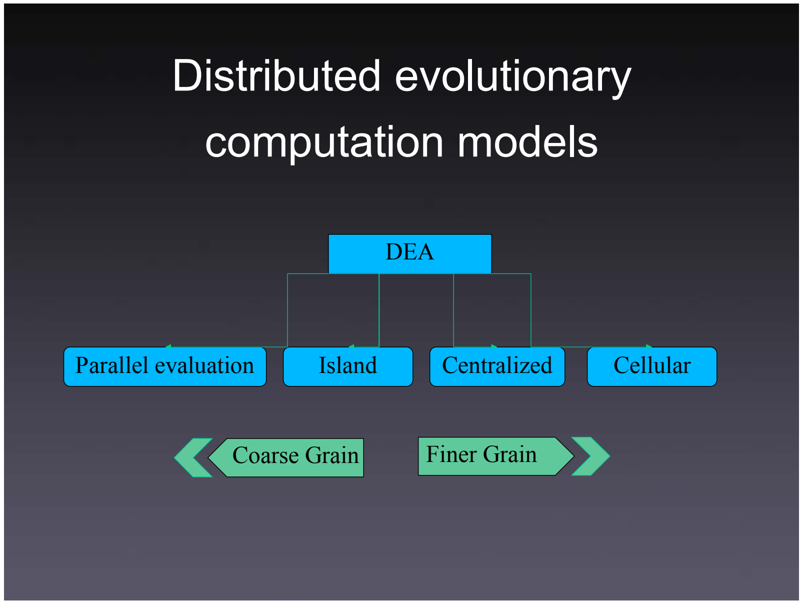
Outline

- Objective
- DEC
- Volunteer computing
- P2P
- Ruby



Our dream

- Milliards of People sharing CPU idle cycles through the Internet
- A crazy scientist designing easily an Evolutionary experiment for fun and/or profit following a simple parallelization model
- The experiment is transparently conducted in a free, highly available HPC platform
- Good Performance, Good Results



Distributed evolutionary computation models

DEA

Parallel evaluation

Island

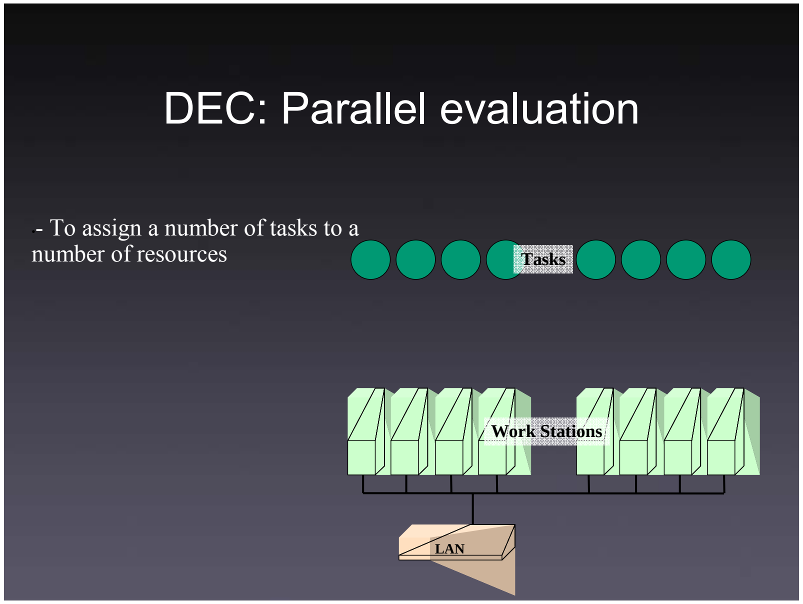
Centralized

Cellular

Coarse Grain

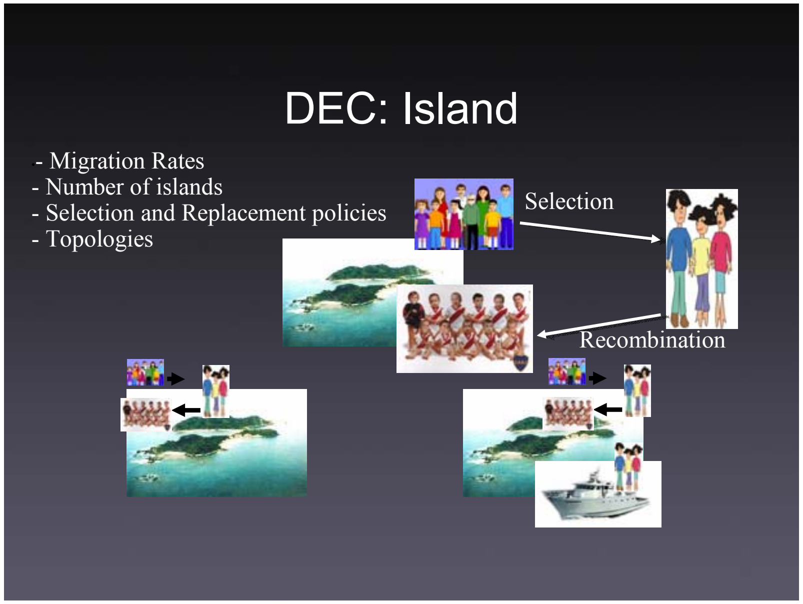
Finer Grain

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
ACM 978-1-59593-698-1/07/0007.



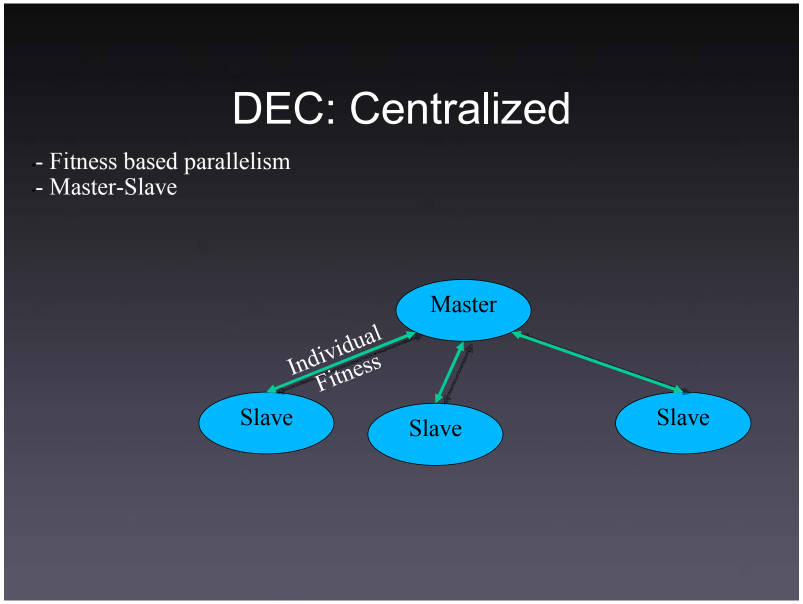
DEC: Parallel evaluation

- To assign a number of tasks to a number of resources



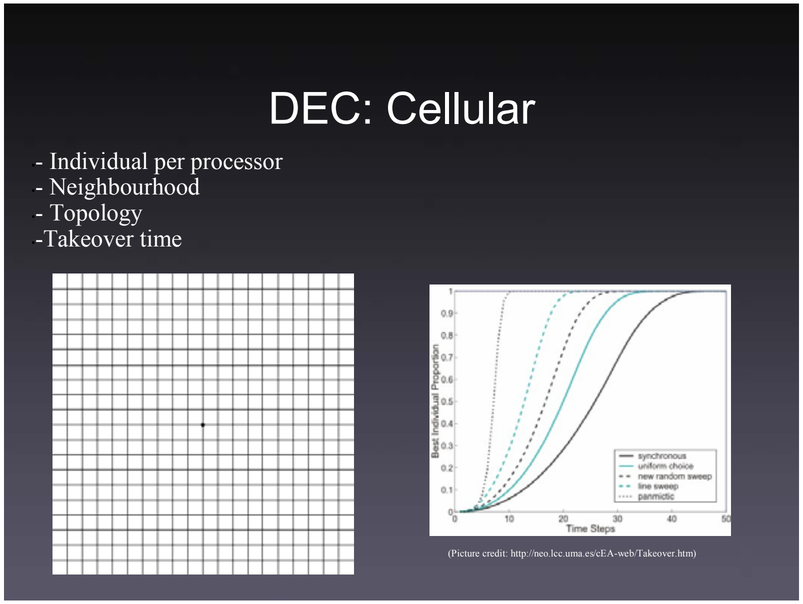
DEC: Island

- Migration Rates
- Number of islands
- Selection and Replacement policies
- Topologies



DEC: Centralized

- Fitness based parallelism
- Master-Slave



DEC: Cellular

- Individual per processor
- Neighbourhood
- Topology
-Takeover time

(Picture credit: http://neo.lcc.uma.es/cEA-web/Takeover.htm)

Issues in DEC

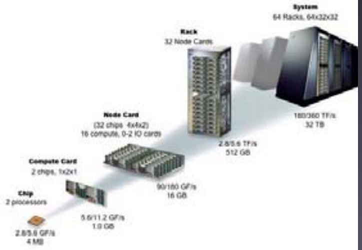
- Scalability
- Performance
- Algorithmic issues
- Resource availability

Implementation Issues

- Volunteer computing
  - P2P Computing
- Grid Computing
- High Performance Computing
- High Throughput Computing

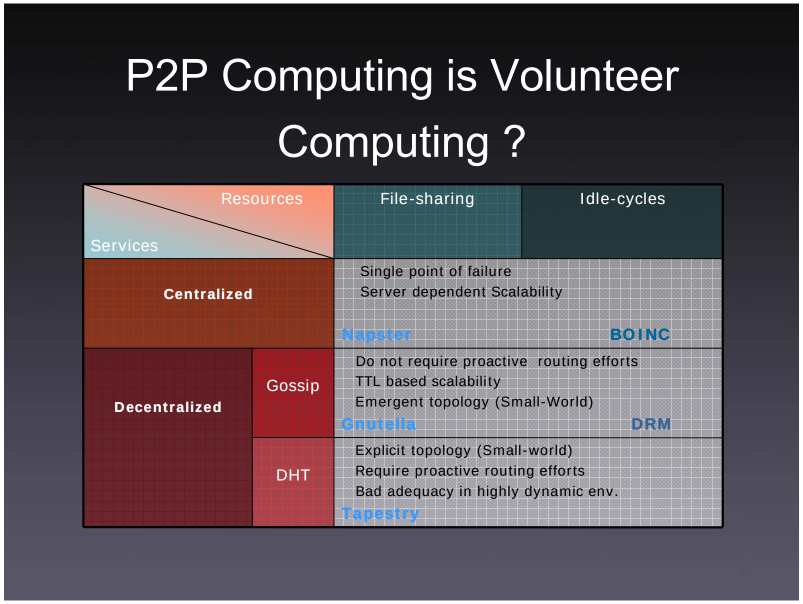
Volunteer Computing (I)

- BlueGene: 1st in Top 500
- 280 Teraflops



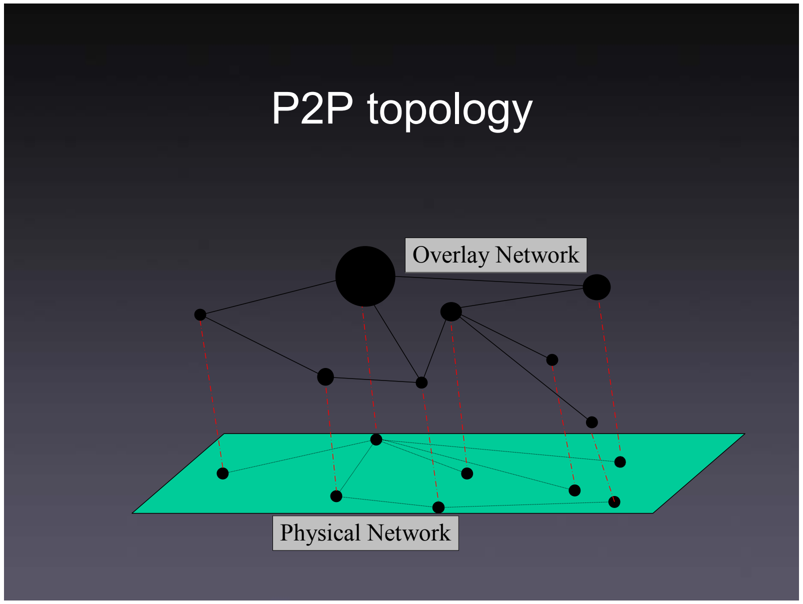
Volunteer Computing (II)

- BOINC (Seti@home, folding@home,...)
- 521 Teraflops
- Folding@home
  - 22000 PlayStation 3
  - 289 Teraflops



P2P Computing is Volunteer Computing ?

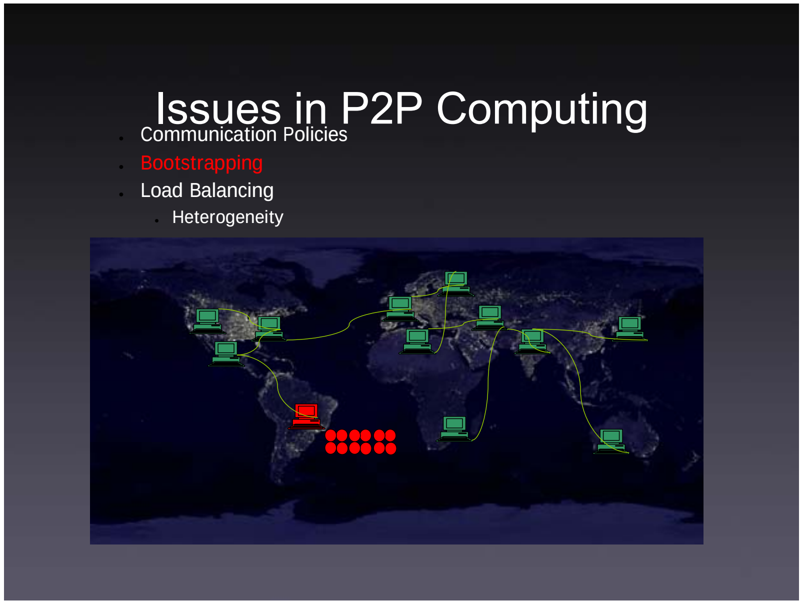
Resources Services		File-sharing	Idle-cycles
Centralized		Single point of failure Server dependent Scalability Napster	BOINC
Decentralized	Gossip	Do not require proactive routing efforts TTL based scalability Emergent topology (Small-World) Gnutella	DRM
	DHT	Explicit topology (Small-world) Require proactive routing efforts Bad adequacy in highly dynamic env. Tapestry	



P2P topology

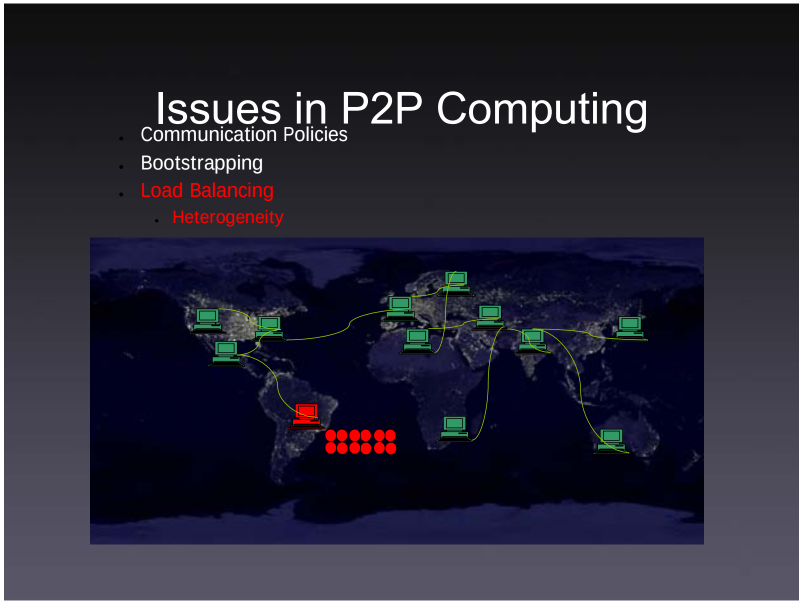
Issues in P2P Computing

- Communication Policies
- Bootstrapping
- Load Balancing
  - Heterogeneity



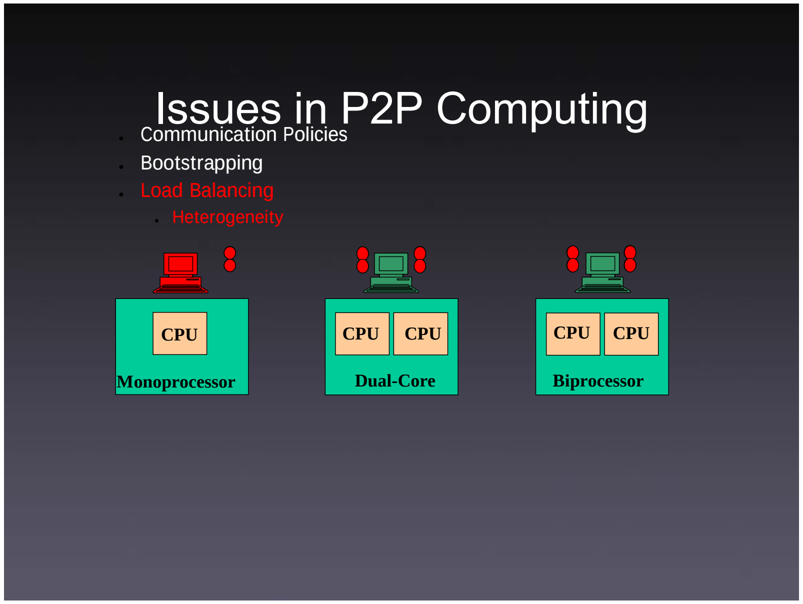
Issues in P2P Computing

- Communication Policies
- Bootstrapping
- Load Balancing
  - Heterogeneity



Issues in P2P Computing

- Communication Policies
- Bootstrapping
- Load Balancing
  - Heterogeneity



Issues in P2P Computing

- Communication Policies
- Bootstrapping
- Load Balancing
  - Heterogeneity

CPU

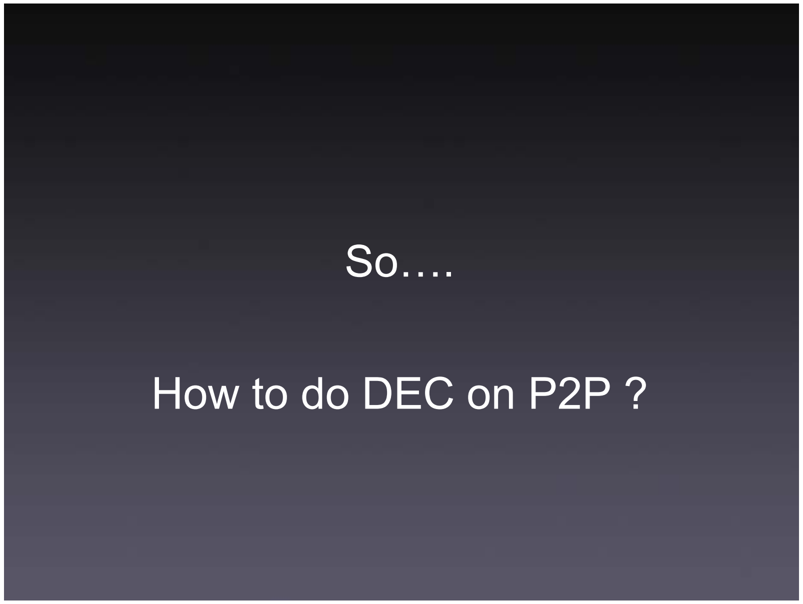
Monoprocessor

CPU CPU

Dual-Core

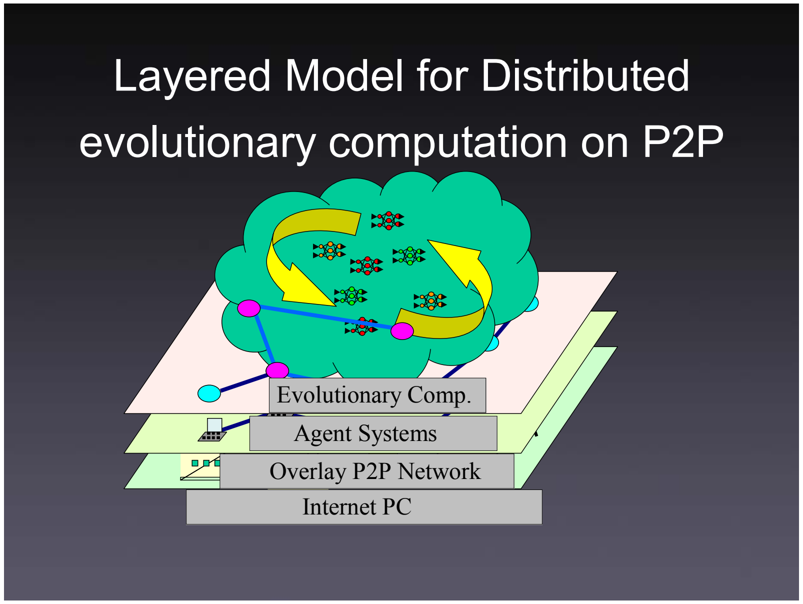
CPU CPU

Biprocessor



So….

How to do DEC on P2P ?



Layered Model for Distributed evolutionary computation on P2P

Evolutionary Comp.

Agent Systems

Overlay P2P Network

Internet PC

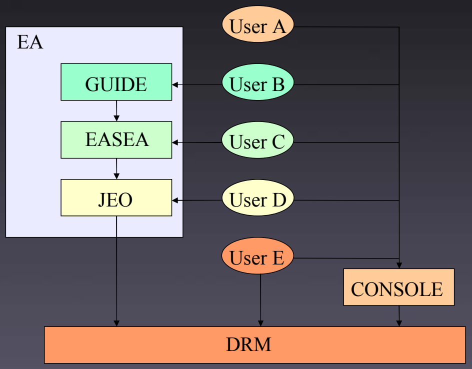
Where can I buy a framework?

- DREAM (http://dr-ea-m.sourceforge.net/)
- JADE
- G2DGA
- ParadisEO
- GPU
- JXTA

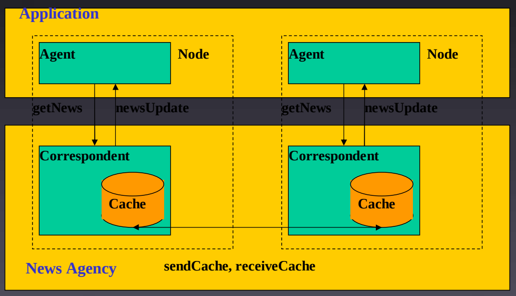
DREAM

- Arenas, M.G. Collet, P., Eiben, A. E., Jelasity, M., Merelo, J. J., Paechter, B., Preuß, M., Schoenauer, M., "A Framework for Distributed Evolutionary Algorithms", Proceedings of PPSN VII, Granada, September 2002.
- M.G. Arenas, B. Dolin, J.J. Merelo, P. A. Castillo, I Fernández De Viana, Marc Schoenauer. JEO: Java Evolving Objects. GECCO 2002. New York. 9-13 July 2002. Morgan Kufmann Publishers..
- Jelasity, M., Preuß, M. and Paechter, B., "A Scaleable and Robust Framework for Distributed Application", Proceedings of the Congress on Evolutionary Computation, Honolulu, pp1540-1545, May 2002

DREAM



Using DRM



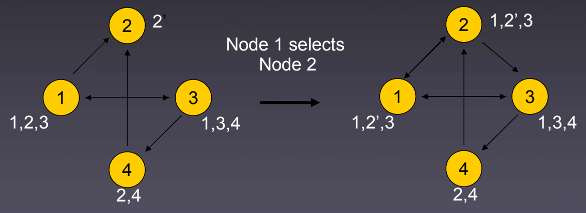
Cache Item

Address	Timestamp	AgentID	Application Data

Using DRM

- Gossip protocol

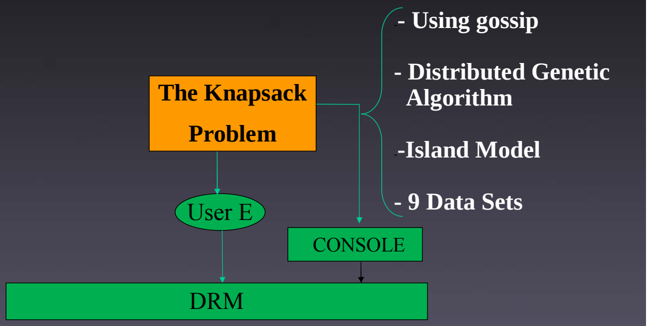
Cache size = 3



JEO+DRM (DREAM)

- Based on Island model
- Gossip just for statistic dissemination
- Direct communication scheme
  - ✗ Predefined number of Island → Bad fault tolerance
  - ✗ What about new resources? → Limited scalability
  - ☑ Good experimental results

Let's experiment with DRM(I)



The Knapsack Problem

User E

CONSOLE

DRM

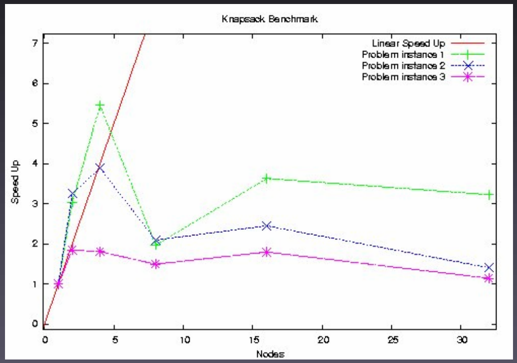
- Using gossip

- Distributed Genetic Algorithm

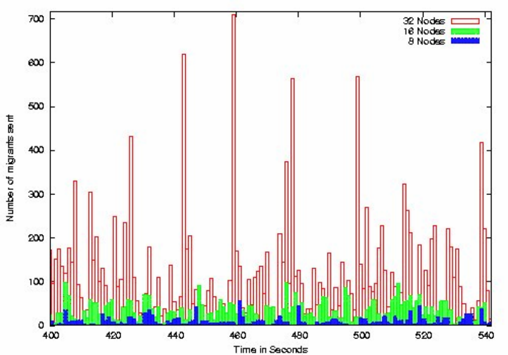
-Island Model

- 9 Data Sets

Let's experiment with DRM(II)



Let’s experiment with DRM(III)



Let’s experiment with DRM(IV)

$$Hops_{avg} \approx N$$

All migrants infects all the nodes

$$\frac{MigrantSent}{generation} \approx 0.1N$$

A migrant infects 10% of the network each generation

Let’s gossip (I)

Gossip

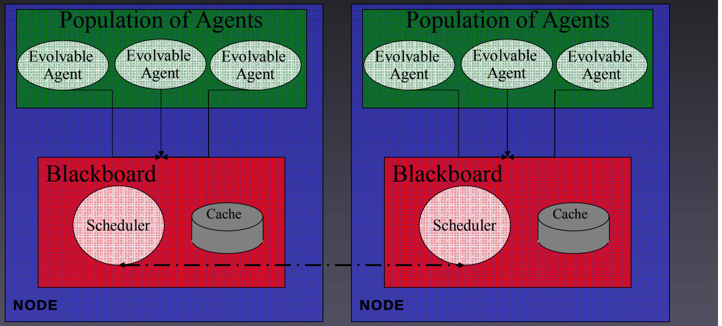
or

Epidemic

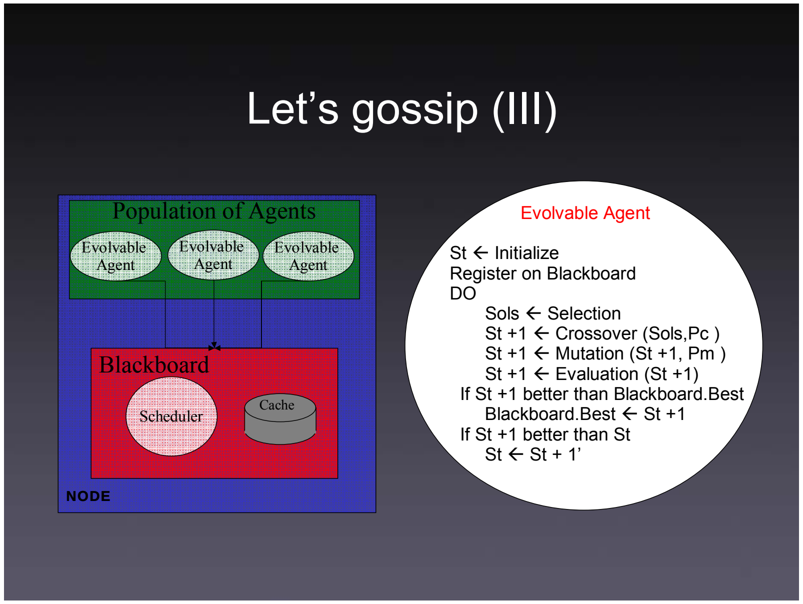
A GOSSIP: CONTRIBUTION

Address	Evaluations	Individual

Let’s gossip (II)



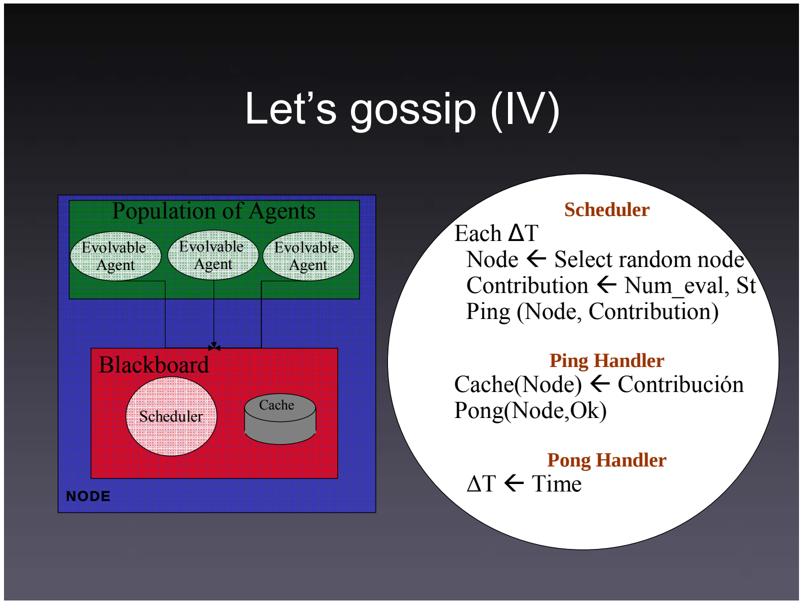
Ping, Pong



Let's gossip (III)

Evolvable Agent

```
St ← Initialize
Register on Blackboard
DO
      Sols ← Selection
      St +1 ← Crossover (Sols,Pc )
      St +1 ← Mutation (St +1, Pm )
      St +1 ← Evaluation (St +1)
   If St +1 better than Blackboard.Best
      Blackboard.Best ← St +1
   If St +1 better than St
      St ← St + 1'
```



Let's gossip (IV)

Scheduler

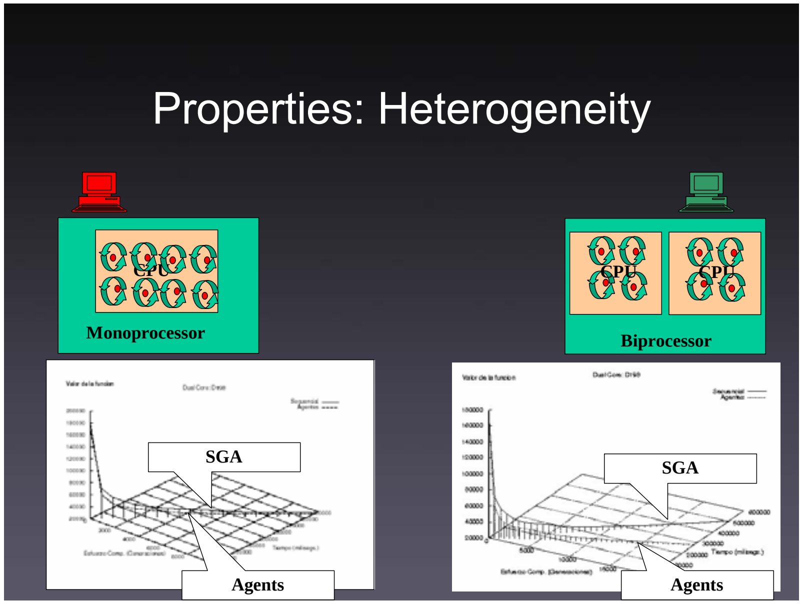
```
Each ΔT
  Node ← Select random node
  Contribution ← Num_eval, St
  Ping (Node, Contribution)
```

Ping Handler

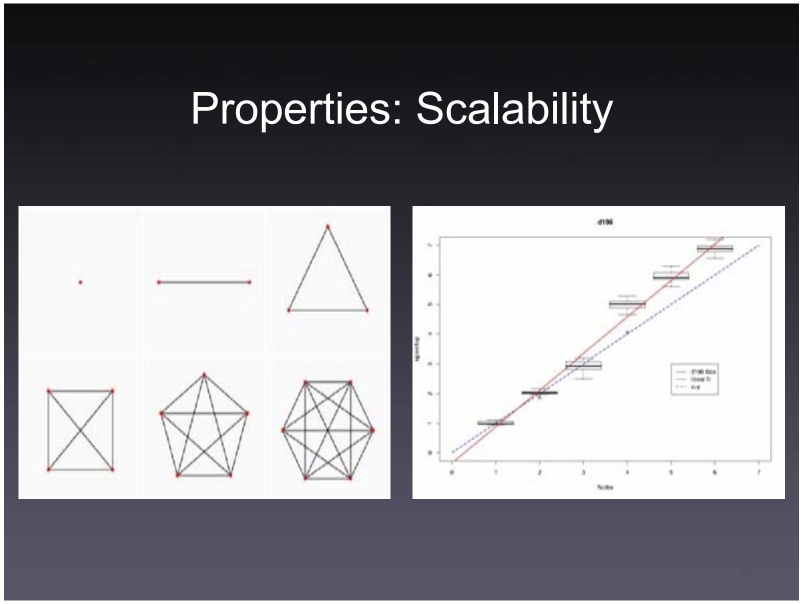
```
Cache(Node) ← Contribución
Pong(Node,Ok)
```

Pong Handler

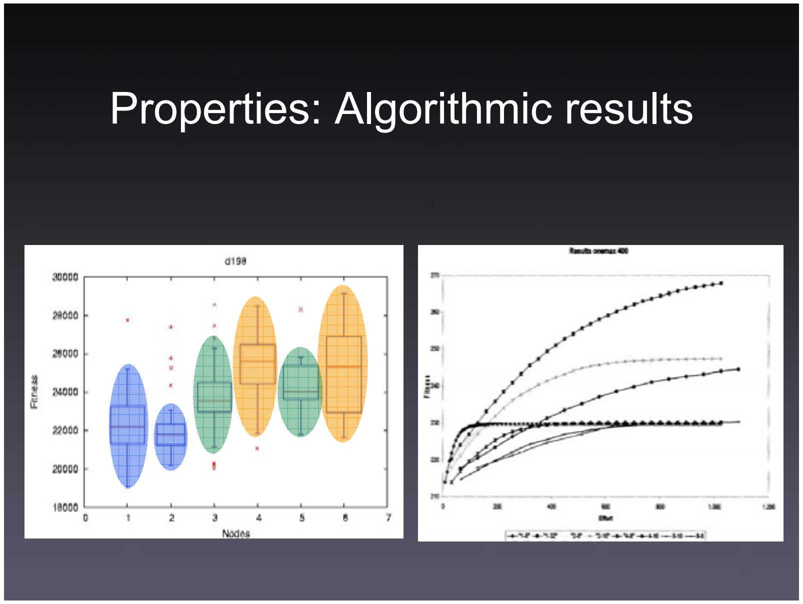
```
  ΔT ← Time
```



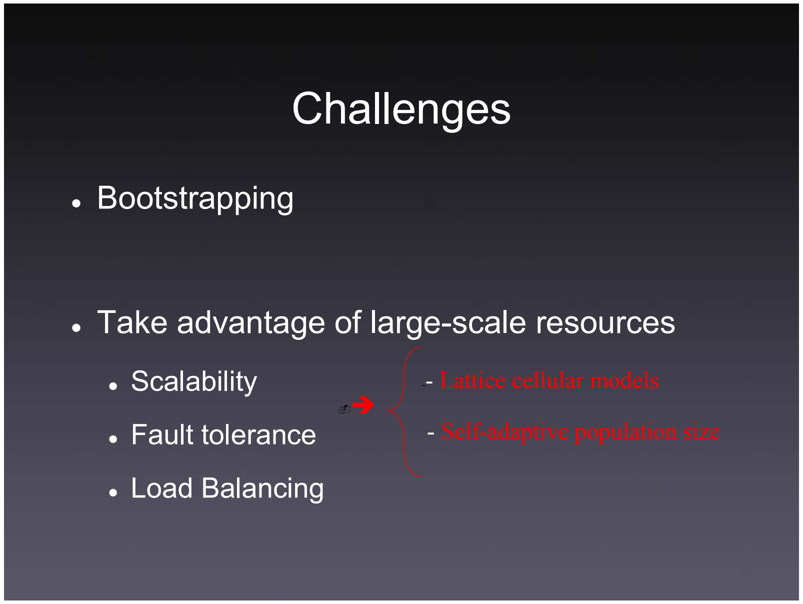
Properties: Heterogeneity



Properties: Scalability



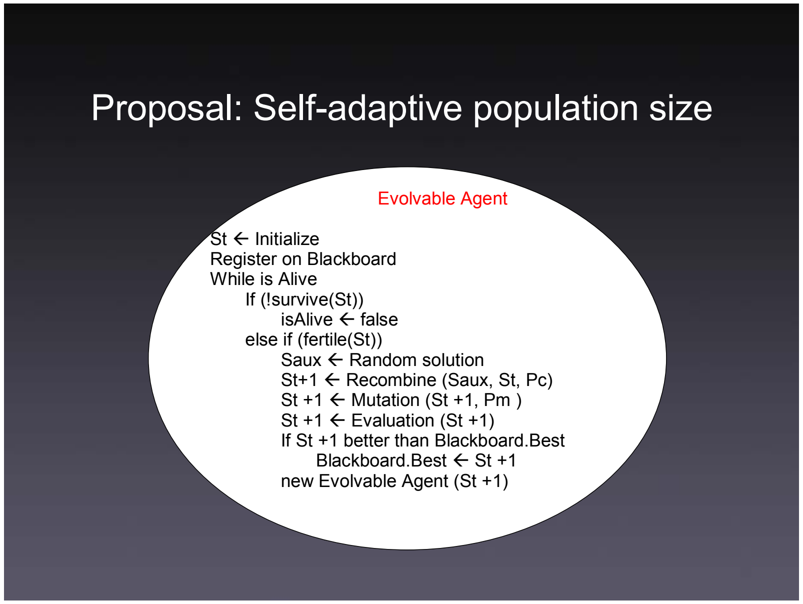
Properties: Algorithmic results



Challenges

- Bootstrapping
- Take advantage of large-scale resources
  - Scalability
  - Fault tolerance
  - Load Balancing

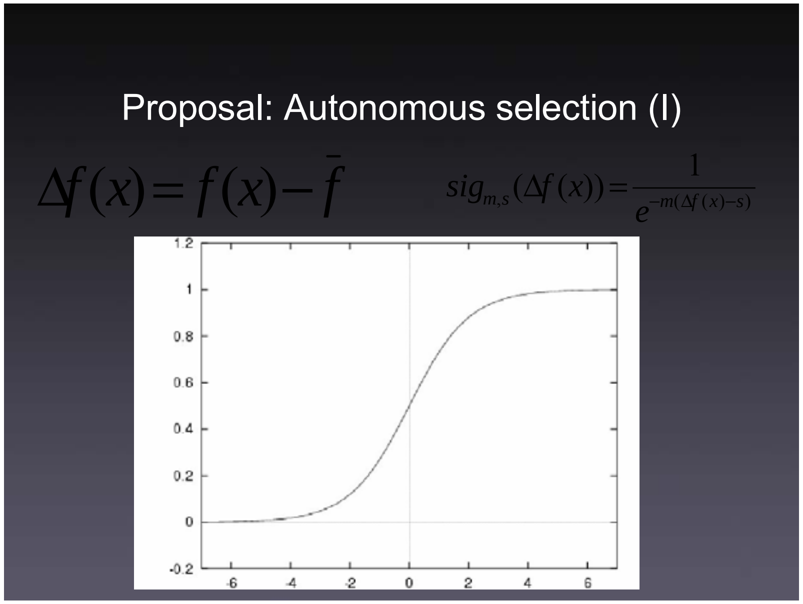
- Lattice cellular models
- Self-adaptive population size



Proposal: Self-adaptive population size

Evolvable Agent

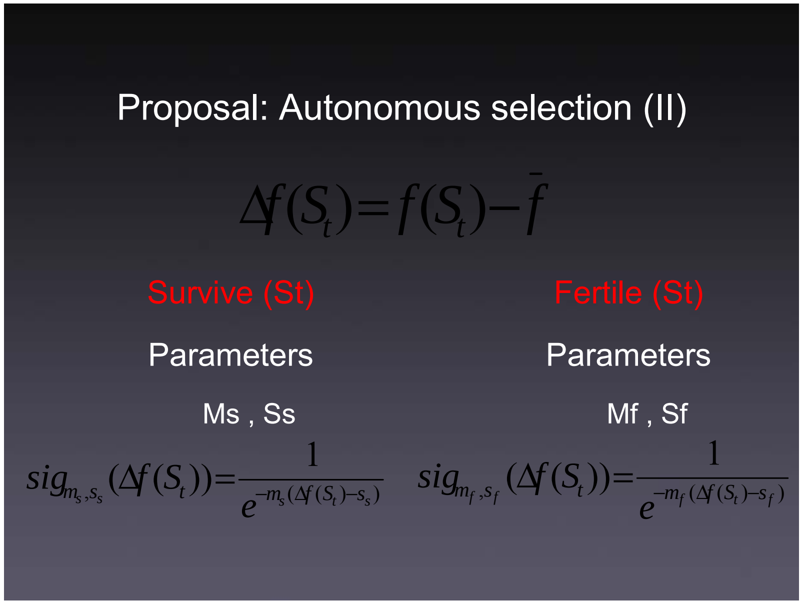
```
St ← Initialize
Register on Blackboard
While is Alive
    If (!survive(St))
        isAlive ← false
    else if (fertile(St))
        Saux ← Random solution
        St+1 ← Recombine (Saux, St, Pc)
        St +1 ← Mutation (St +1, Pm )
        St +1 ← Evaluation (St +1)
        If St +1 better than Blackboard.Best
            Blackboard.Best ← St +1
        new Evolvable Agent (St +1)
```



Proposal: Autonomous selection (I)

$$\Delta f(x) = f(x) - \bar{f}$$

$$sig_{m,s}(\Delta f(x)) = \frac{1}{e^{-m(\Delta f(x)-s)}}$$



Proposal: Autonomous selection (II)

$$\Delta f(S_t)=f(S_t)-\bar{f}$$

Survive (St)

Parameters

Ms , Ss

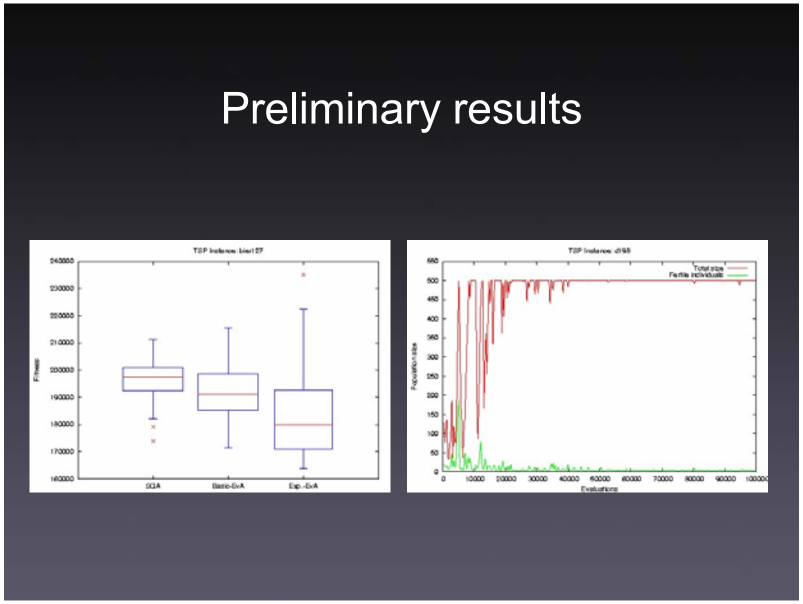
$$sig_{m_s,s_s}(\Delta f(S_t))=\frac{1}{e^{-m_s(\Delta f(S_t)-s_s)}}$$

Fertile (St)

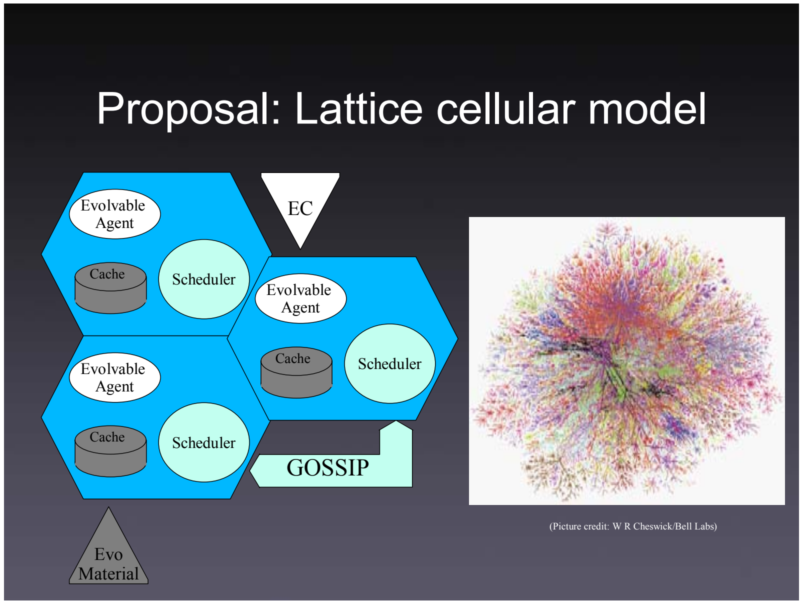
Parameters

Mf , Sf

$$sig_{m_f,s_f}(\Delta f(S_t))=\frac{1}{e^{-m_f(\Delta f(S_t)-s_f)}}$$

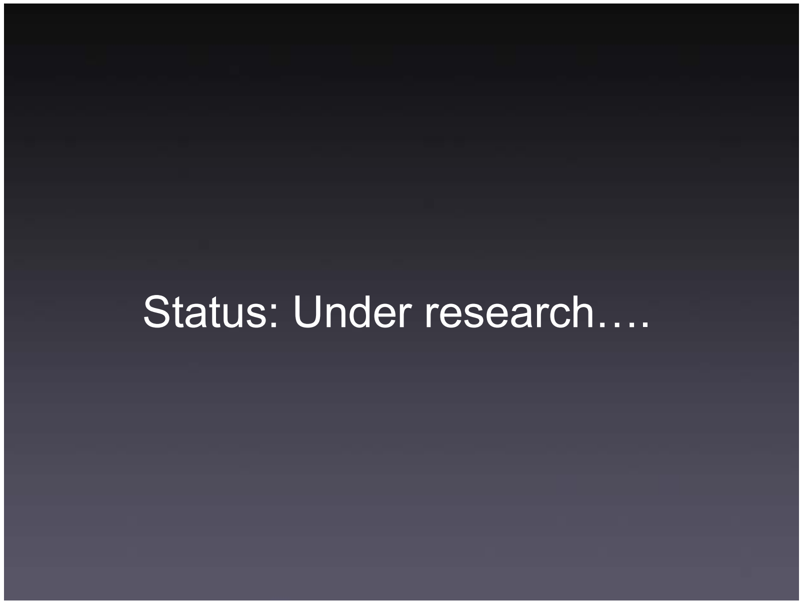


Preliminary results

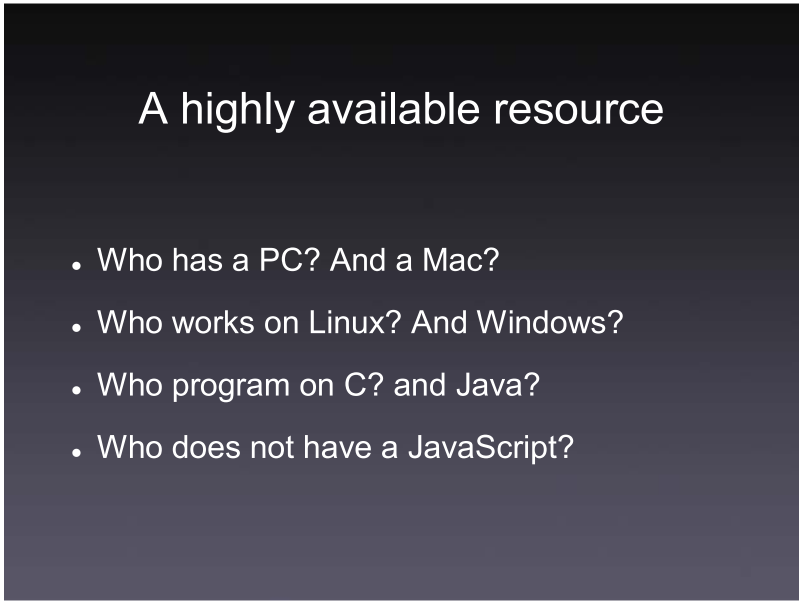


Proposal: Lattice cellular model

(Picture credit: W R Cheswick/Bell Labs)



Status: Under research….



A highly available resource

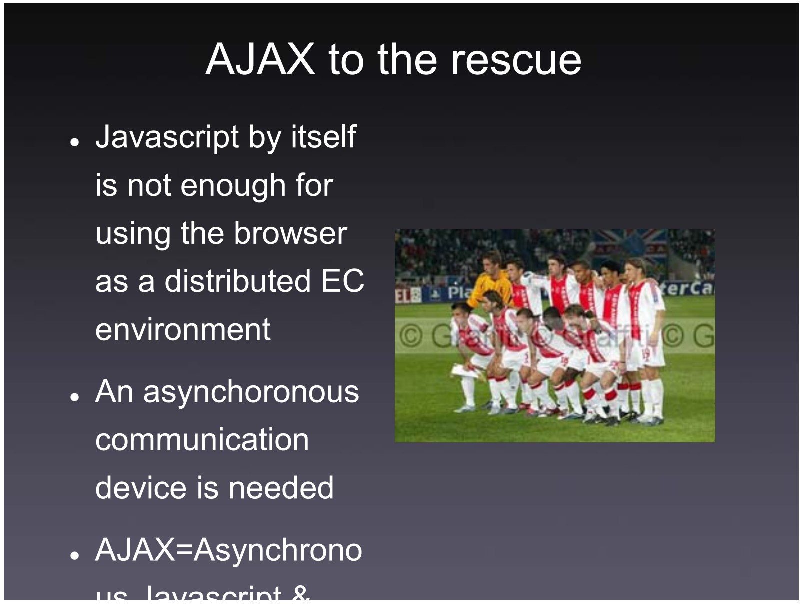
- Who has a PC? And a Mac?
- Who works on Linux? And Windows?
- Who program on C? and Java?
- Who does not have a JavaScript?

Let’s have fun: We’ve found a Gem

And we know how to use it

Browswers are everywhere

- And browsers are actually virtual machines that can be used for evolutionary computation.
- Javascript is in every browser
    - And in every machine



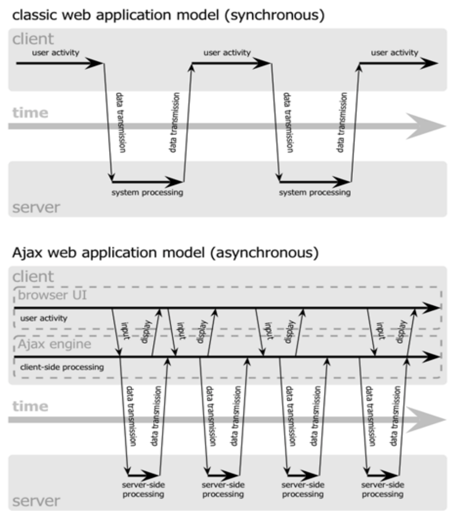
AJAX to the rescue

- Javascript by itself is not enough for using the browser as a distributed EC environment
- An asynchoronous communication device is needed
- AJAX=Asynchronous Javascript &

AJAX for everyone

- Most browsers include Javascript/ECMAscript
- The object model is also compatible.
- `XMLHttpRequest` is a must.

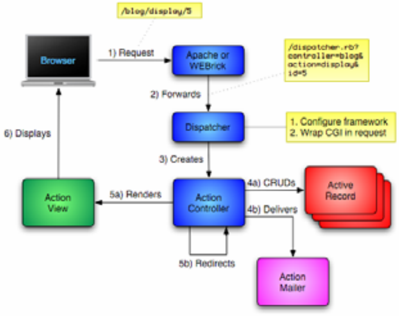
Allright, how does it work?



Slow train coming

- **Ruby on Rails is an agile development environment based on the MVC paradigm, the Ruby language, AJAX a**

MySQL



What is DCoR?

- *Distributed computation on rails is a distributed computation system, geared for evolutionary computation, based on Ruby on Rails.*
- The distribution model is client/server
  - But servers can be linked.
- It's still on the *proof of concept phase.*
  - *Testing for browser performance and other parameters.*

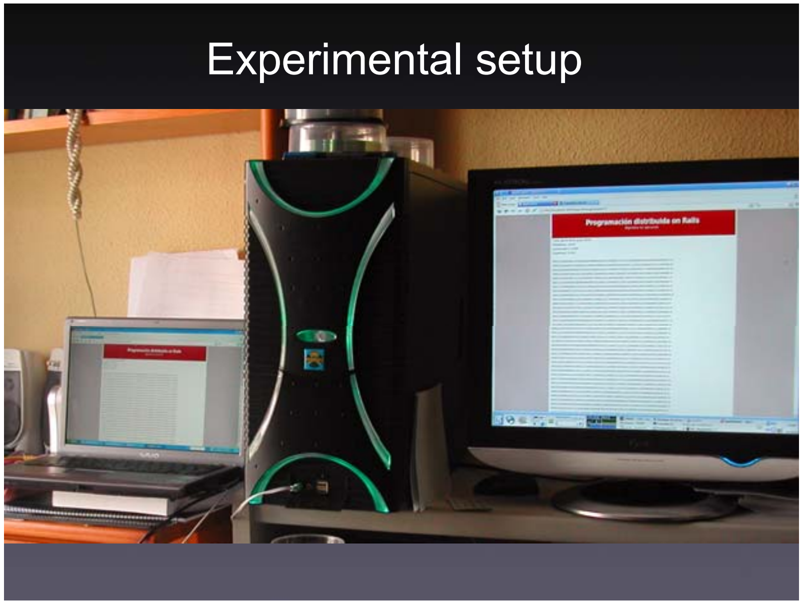
Wanted: cool logo

Don't try this at home

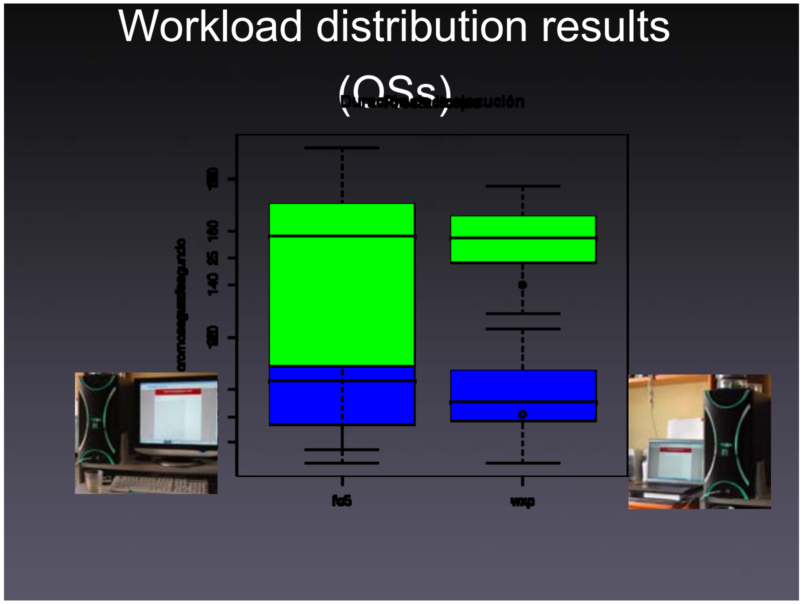
- Royal Road problem
  - Usual testbed for EC problems
  - A way to test integer performance.

1 1 1 → =3

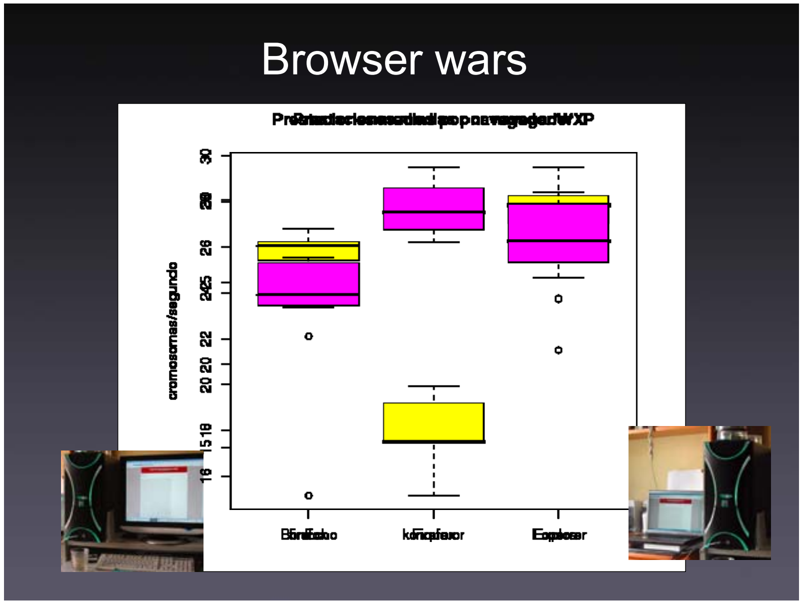
1 0 1 → =0



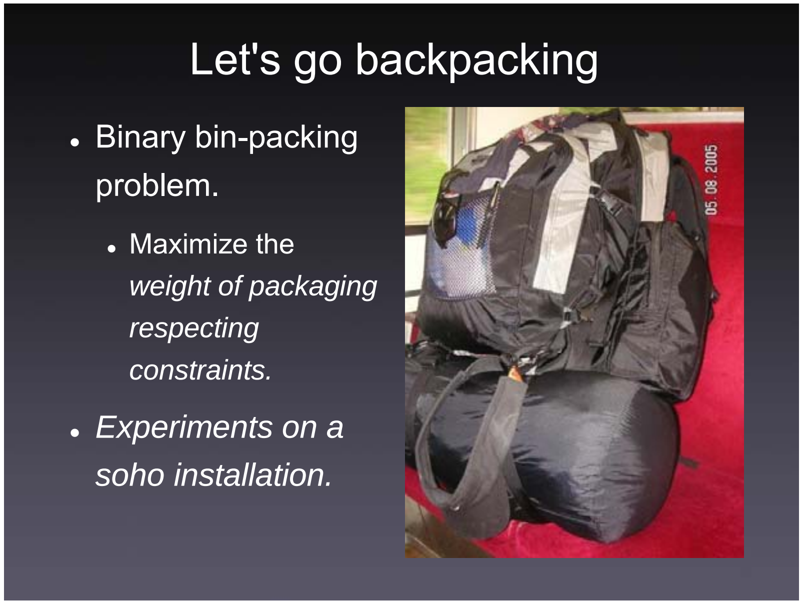
Experimental setup



Workload distribution results (OSs)

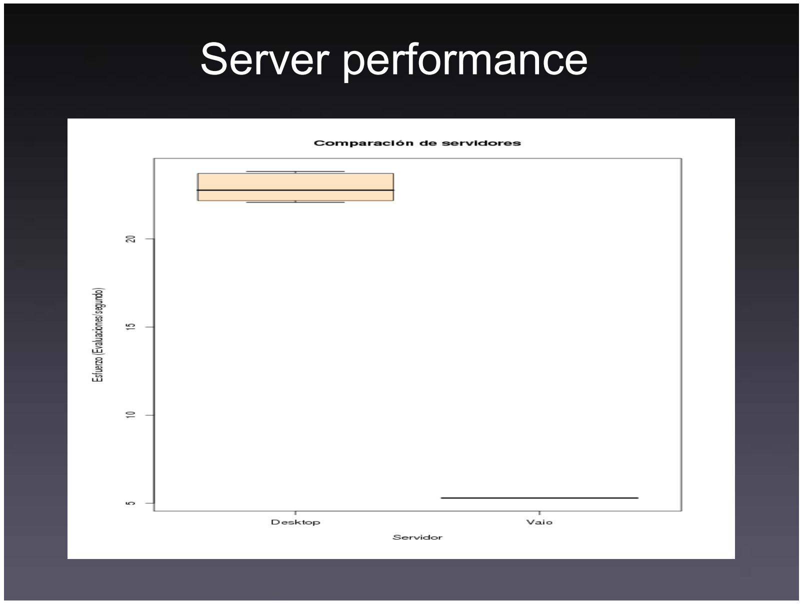


Browser wars

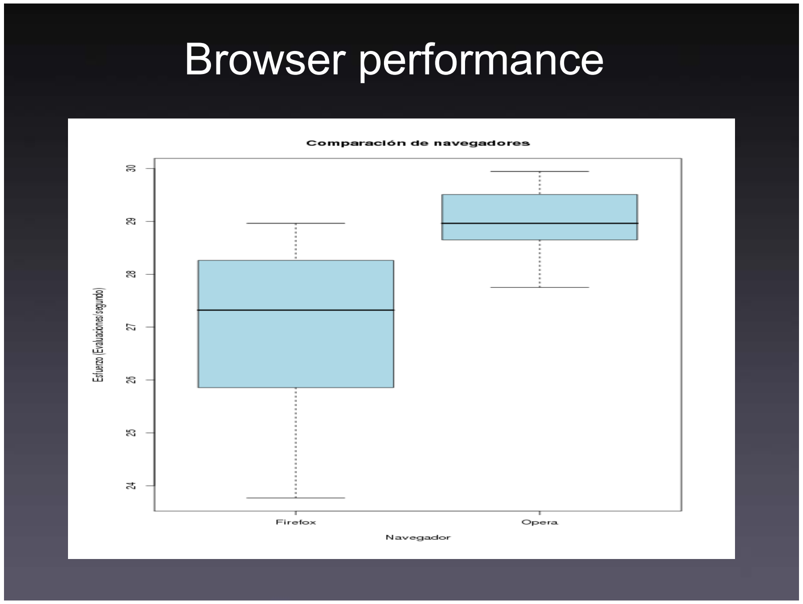


Let's go backpacking

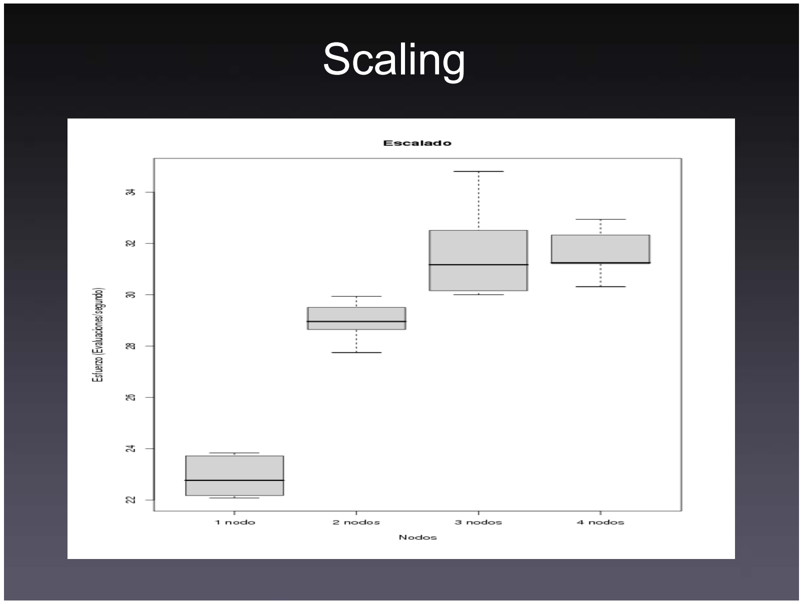
- Binary bin-packing problem.
  - Maximize the *weight of packaging respecting constraints.*
- *Experiments on a soho installation.*



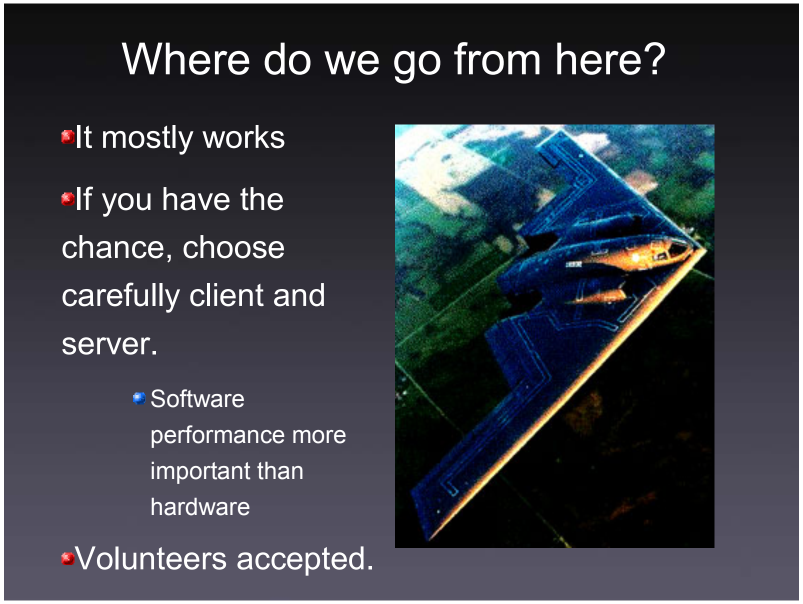
Server performance



Browser performance

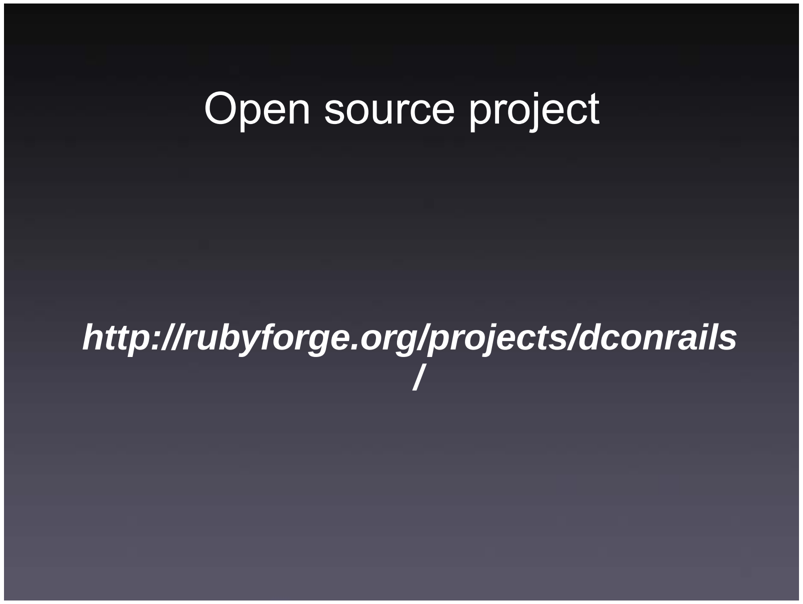


Scaling



Where do we go from here?

- It mostly works
- If you have the chance, choose carefully client and server.
  - Software performance more important than hardware
- Volunteers accepted.



Open source project

http://rubyforge.org/projects/dconrails/

That's all

Thank you very much