# Communication Theory in the Cloud

**Matthew C. Valenti** [1]

West Virginia University

October 30, 2009

[1] This work supported by the National Science Foundation under award CNS-0750821.

# Outline

1. Motivation
2. How to Run Matlab on a Grid
3. A Cloud Computing Approach
4. Evolving on a Grid
5. Conclusions

# Monte Carlo Simulation Ten Years Ago

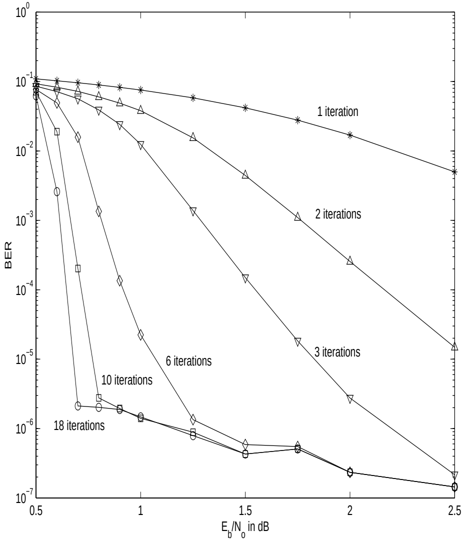


Figure 3.3 from my 1999 dissertation:

- "Iterative detection and decoding for wireless communications."
- Simulated the original turbo code of Berrou et al (1993) down to a BER of $10^{-7}$.
- Simulation written using C-mex and ran on 266 MHz processor.
- Took about a month to generate this figure!

# Today's World

- Monte-Carlo simulation continues to be an important aspect of communication theory.
- Processing capacity has increased, but so has the complexity of the systems that must be simulated.
- Other compute-intensive algorithms have been successfully applied to the analysis and design of systems.
    - Density-evolution.
    - Genetic algorithms.
- **Access to computing power is an advantage to groups working on modern communication theory.**
- The demand for computing resources can now be met by *utility*, *grid*, and *cloud* computing.

# Utility and Cloud Computing

**Utility computing** providers rent capacity on computing resources that they maintain.

- Metered computing: analogous to electric power.
- Resources often virtualized and shared by multiple tenants.
- Example: *Amazon Elastic Compute Cloud* ($2.04 USD/day).

**Cloud computing** not only provides raw computing power, but also hosts the applications that use the power.

- Applications usually accessed via a web browser.
- User data typically stored on provider's filesystems.
- Underlying computing infrastructure concealed from user.
- Key example: *gmail*.

# Cluster, Grid, and Community Computing

A **cluster** computer is a collection of tightly coupled computing servers.

- Usually co-located.

A computing **grid** is a distributed collection of computing servers.

- While the servers may be dedicated resources, they could be borrowed from idle desktop computers.
- Specialized software is needed to aggregate CPU power (e.g. *Frontier*).

**Community computing** projects assemble a grid of donated CPU resources using volunteers idle cycles.

- Examples: SEI@home, Folding@home (Driven by *BOINC* software).
- Screensaver (windows) or low-priority process (linux).

# Our Initial Work on Grid Computing

Starting in 2005, we began partnering with Parabon Computation, Inc.

- Developer of the *Frontier* grid computing platform.

The Frontier client was installed on over 2,000 desktop computers throughout the state of WV.

- 31 from one of our general-purpose student computing lab were set aside for my research group's experimentation.
- 11 windows and 20 linux machines.

We developed a methodology for executing Matlab code on the grid.

- Interface was through Matlab commands (no web interface, yet).

# The Matlab Compiler

A key enabling technology was the Matlab compiler.

- *mcc*.
- Translates Matlab to C, then compiles to executable.

We ran the compiled Matlab code on the grid.

- Benefit: No Matlab license needed on the compute engines.
- Issue: Executable is architecture specific.

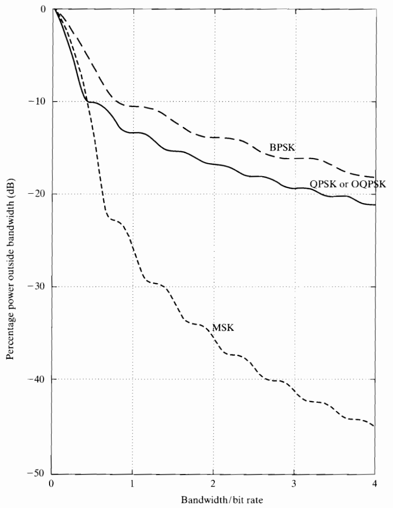
In addition to the compiled executables, the Matlab Component Runtime (MCR) must be installed on each machine.

- Around 400-500 MB.
- Must be same version as the compiler.

# Initial Application: Optimization of CPM Modulation I/II

**Continuous-phase modulation** (CPM) is a class of modulation characterized by a continuous phase transition from one symbol interval to the next.

- Constant amplitude (unity PAPR) for efficient amplification.
- Low spectral sidelobes for reduced ACI.
- Suitable for noncoherent reception.



# Initial Application: Optimization of CPM Modulation II/II

We considered coded CPM modulation, and optimized:

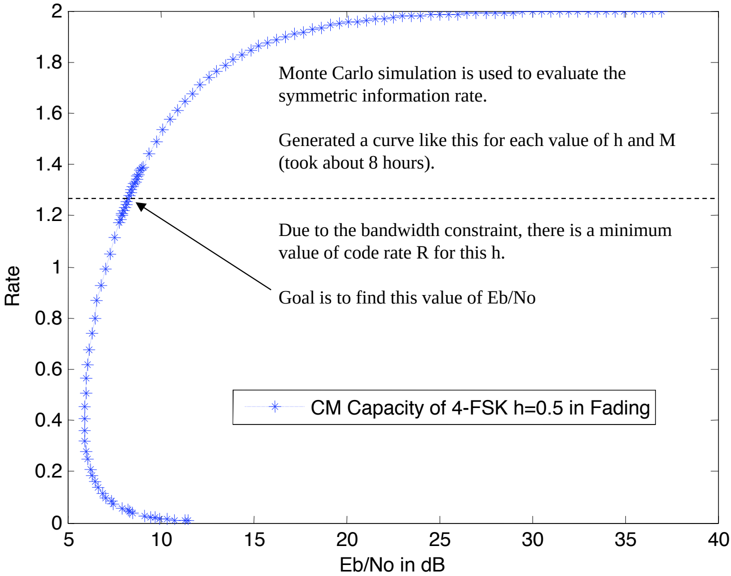
- R: rate of the code.
- h: Modulation index.
- M: Alphabet (e.g. how many tones).

Criteria was the "symmetric information rate".

- Mutual information with uniformly distributed modulator inputs.
- Information-theoretic limit on performance.

Given a bandwidth constraint and value of M, we determined value of (h,R) that minimized the required SNR ($\mathcal{E}_b/N_0$).

# Simulation for a Single $h$



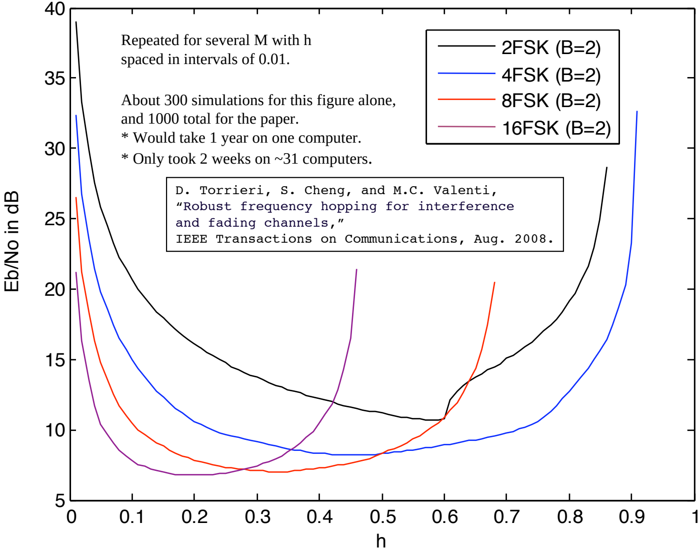
Monte Carlo simulation is used to evaluate the symmetric information rate.

Generated a curve like this for each value of h and M (took about 8 hours).

Due to the bandwidth constraint, there is a minimum value of code rate R for this h.

Goal is to find this value of Eb/No

# Simulation of Many $h$

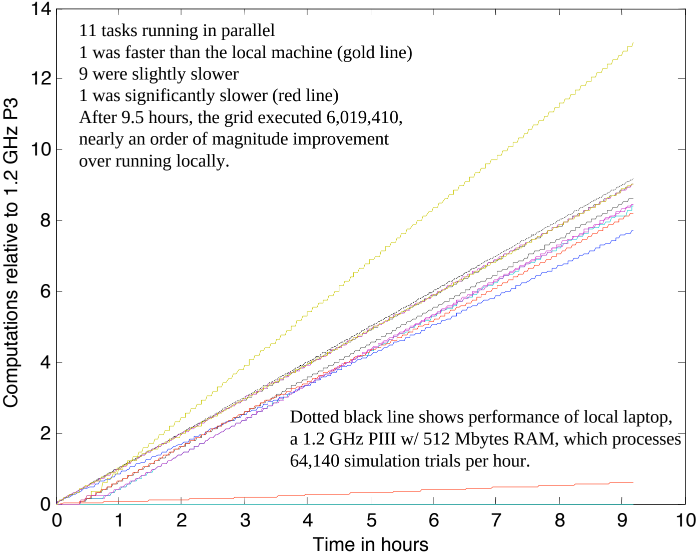


Repeated for several M with h spaced in intervals of 0.01.

About 300 simulations for this figure alone, and 1000 total for the paper.
* Would take 1 year on one computer.
* Only took 2 weeks on ~31 computers.

D. Torrieri, S. Cheng, and M.C. Valenti, "Robust frequency hopping for interference and fading channels," IEEE Transactions on Communications, Aug. 2008.

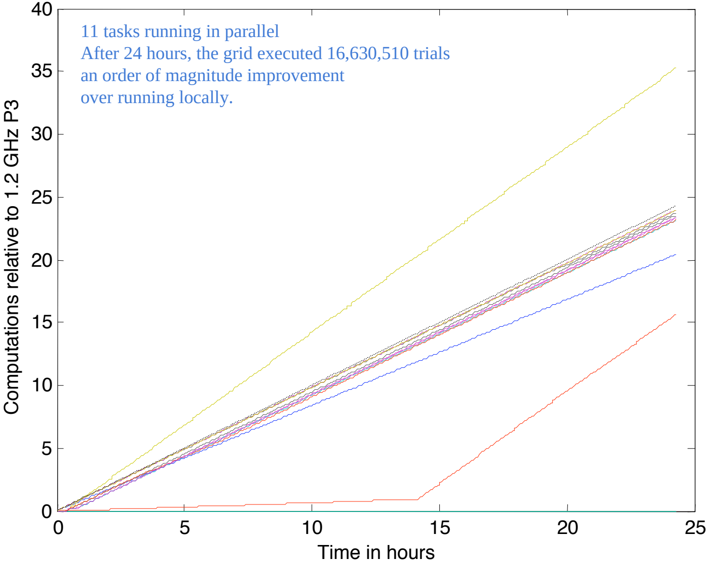
# Runtime Comparison (9.5 hours)



11 tasks running in parallel
1 was faster than the local machine (gold line)
9 were slightly slower
1 was significantly slower (red line)
After 9.5 hours, the grid executed 6,019,410,
nearly an order of magnitude improvement
over running locally.

Dotted black line shows performance of local laptop,
a 1.2 GHz PIII w/ 512 Mbytes RAM, which processes
64,140 simulation trials per hour.

# Runtime Comparison (24 hours)



11 tasks running in parallel
After 24 hours, the grid executed 16,630,510 trials
an order of magnitude improvement
over running locally.

# A Community Computing Resource

In 2007, we received NSF funding to make our work more accessible to the research community through the following enhancements:

- Web-accessibility:
  - A web-interface allowing global access.
- True community computing:
  - Users of the resource expected to contribute CPU cycles in return for the privilege of running on the grid.
  - Will allow the grid to organically grow and scale to the user population.
  - No need to purchase computing power from utilities.
- Parallelization of individual jobs:
  - Split each Monte Carlo simulation into parallel tasks.
- Open-source simulation code:
  - A Matlab toolbox for performing common tasks, such as simulation of turbo and LDPC codes.

# Coded Modulation Library I/II

Coded Modulation Library (CML)

- Developed at WVU.
- First released in 2005.
- Runs in Matlab, but uses lots of c-mex.
- Free software (licensed under lesser GPL).
- Download from www.iterativesolutions.com (version 1.10).

# Coded Modulation Library II/II

Features:

- Modulation: PSK, QAM, APSK, CPM (CPFSK).
- Coding: convolutional, turbo, BTC, LDPC. Hybrid-ARQ.
- Information theoretic bounds (channel capacity; outage probability).
- Support for standardized error control codes:
    - Cellular: WCDMA, HSDPA, LTE, cdma2000.
    - Wireless LAN/MAN: 802.11a/g, 802.16 (Wimax) .
    - Satellite: DVB-RCS, DVB-S2.

# CML Version 2

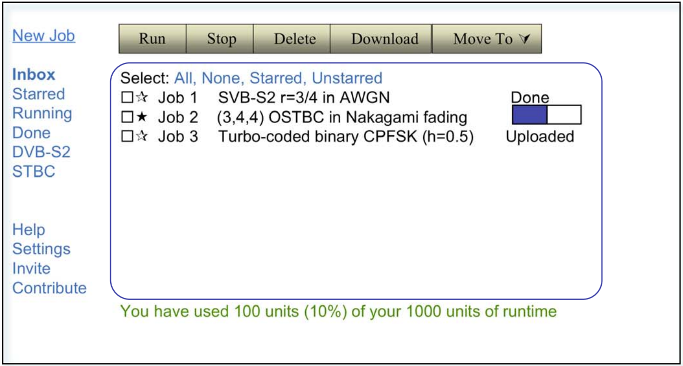
CML 2.0 is now under development.

- Object-oriented.
    - Makes use of Matlab's growing support for creating custom classes.
    - OO makes it easier to add support for new functionality.
- Will include support for STBC, OFDM, and EXIT curves.
- Hosted as a Google Code project.
    - http://code.google.com/p/iscml/
    - Encourages contribution by the community.
    - Uses subversion for version control.

# Parallelizing CML Simulations on the Grid

- In our earlier work, we wanted to run many simulations, so we assigned one simulation per node on the grid.
- However, Monte Carlo simulations are embarrassingly parallelizable.
  - Each Monte Carlo job can be split into multiple tasks.
  - Each task runs as many Monte Carlo trials as it can for 5 minutes or until a maximum number of trials is reached.
  - "Simulation unit".
- The Frontier Application aggregates the data returned from the compute engines.

# The User Perspective



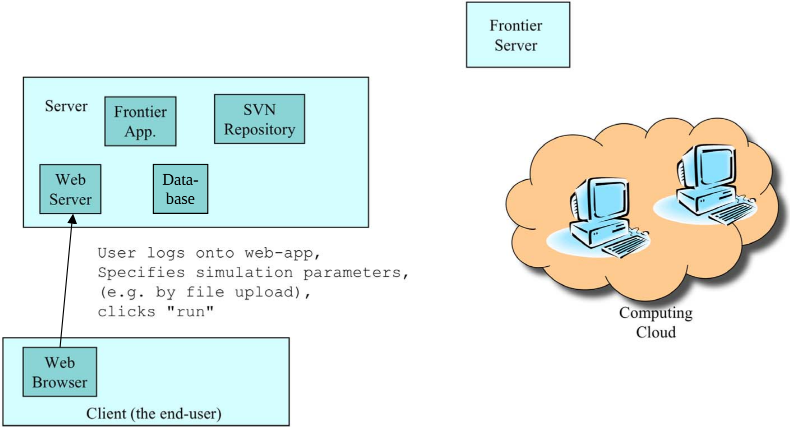
Select: All, None, Starred, Unstarred

| | | | |
|---|---|---|---|
| □☆ | Job 1 | SVB-S2 r=3/4 in AWGN | Done |
| □★ | Job 2 | (3,4,4) OSTBC in Nakagami fading | |
| □☆ | Job 3 | Turbo-coded binary CPFSK (h=0.5) | Uploaded |

You have used 100 units (10%) of your 1000 units of runtime

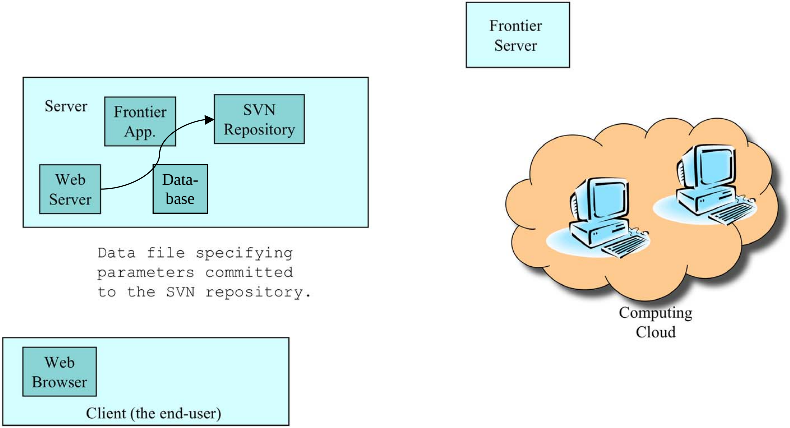
The web interface is developed using the Google Web Toolkit (GWT), which is the same technology used to power gmail.

# How It Works



```
User logs onto web-app,
Specifies simulation parameters,
(e.g. by file upload),
clicks "run"
```

# How It Works



Frontier Server

Server

Frontier App.

SVN Repository

Web Server

Data- base

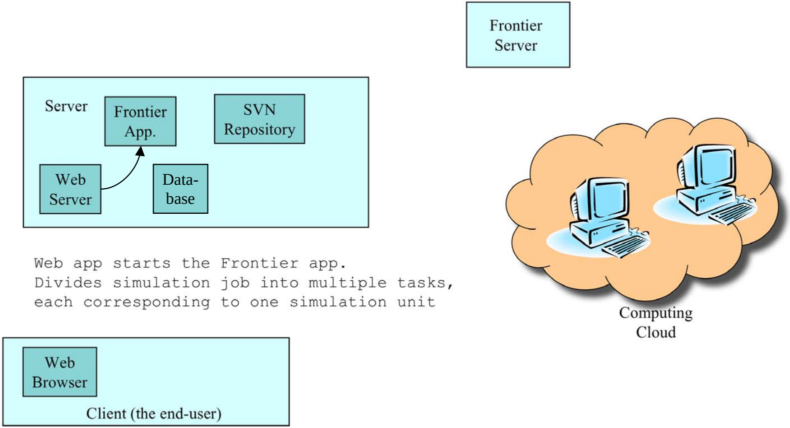
Data file specifying parameters committed to the SVN repository.

Computing Cloud

Web Browser

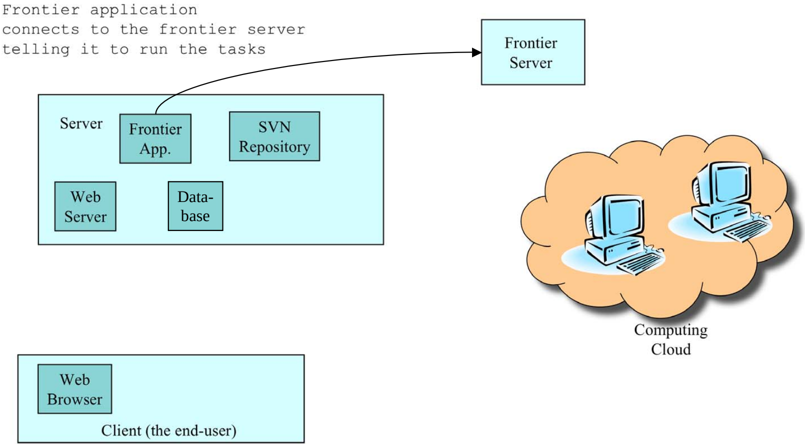
Client (the end-user)

# How It Works



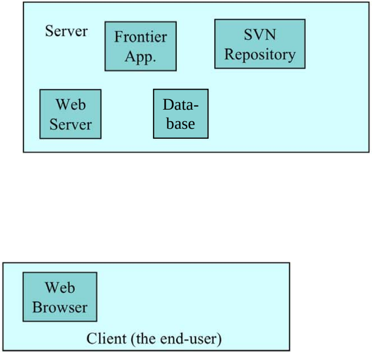
```
Web app starts the Frontier app.
Divides simulation job into multiple tasks,
each corresponding to one simulation unit
```

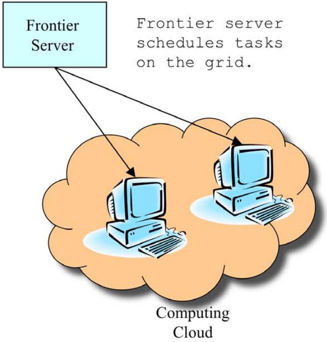
# How It Works



Frontier application connects to the frontier server telling it to run the tasks

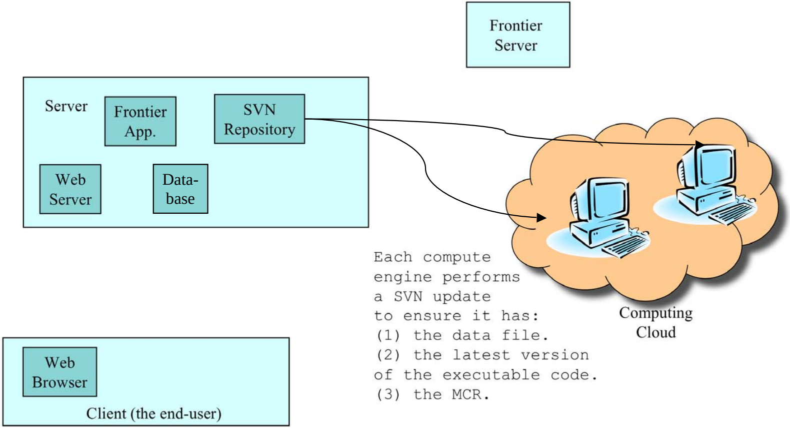
# How It Works





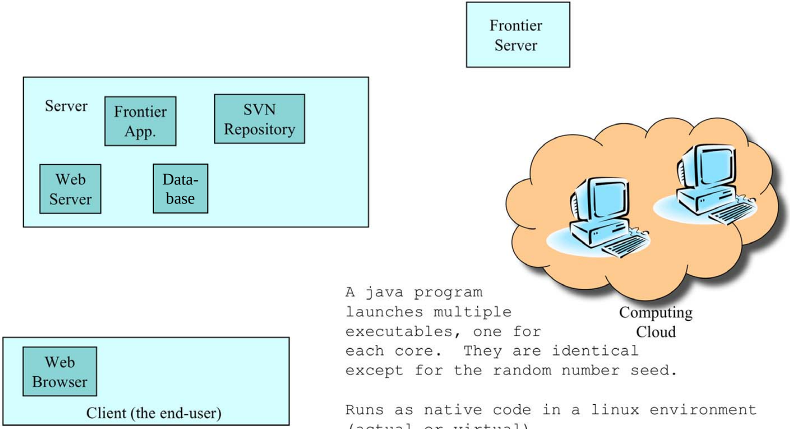
Frontier server schedules tasks on the grid.

# How It Works



Each compute engine performs a SVN update to ensure it has:
(1) the data file.
(2) the latest version of the executable code.
(3) the MCR.

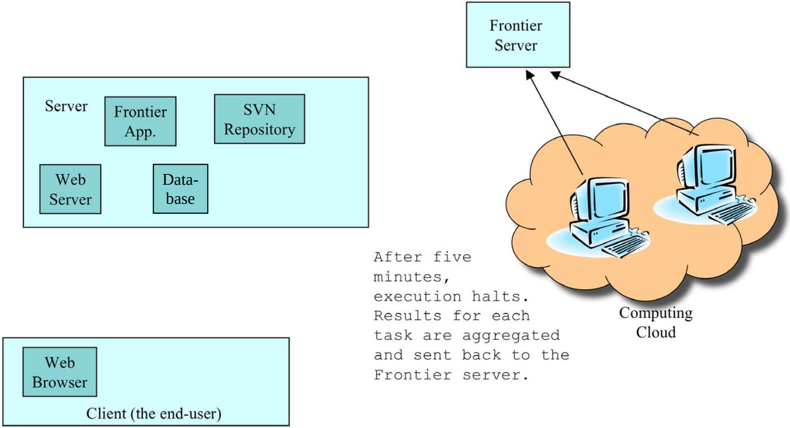
# How It Works



A java program launches multiple executables, one for each core. They are identical except for the random number seed.

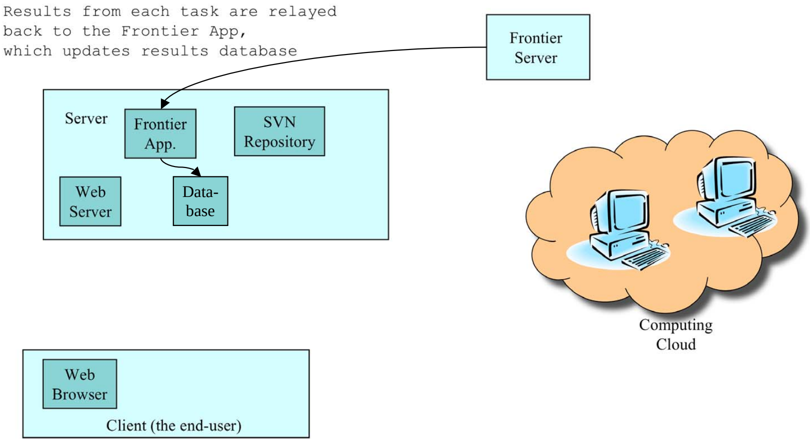
Runs as native code in a linux environment (actual or virtual).

# How It Works

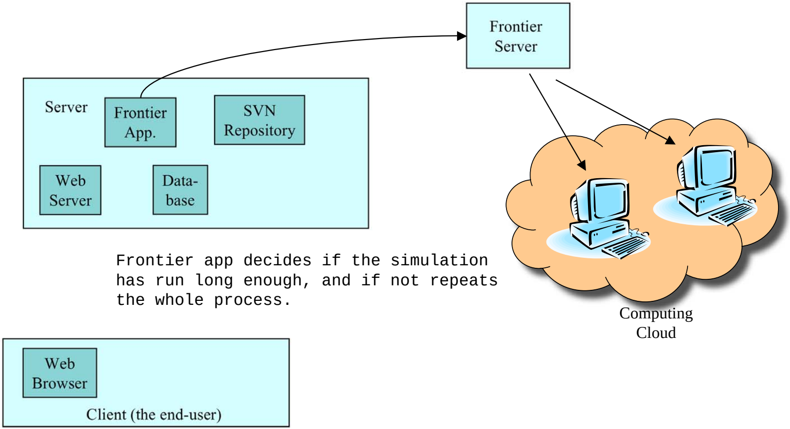


After five minutes, execution halts. Results for each task are aggregated and sent back to the Frontier server.

# How It Works


Results from each task are relayed back to the Frontier App, which updates results database

# How It Works



Frontier Server

Server

Frontier App.

SVN Repository

Web Server

Data-base

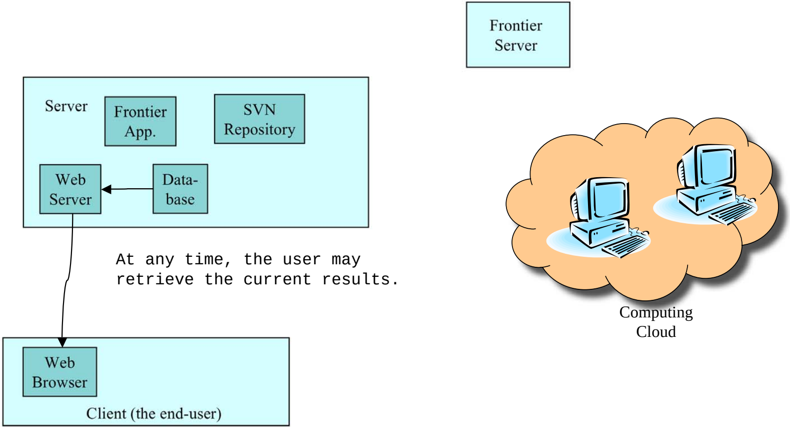
Frontier app decides if the simulation has run long enough, and if not repeats the whole process.

Computing Cloud

Web Browser

Client (the end-user)

# How It Works



Frontier Server

Server

Frontier App.

SVN Repository

Web Server

Data-base

At any time, the user may retrieve the current results.

Web Browser

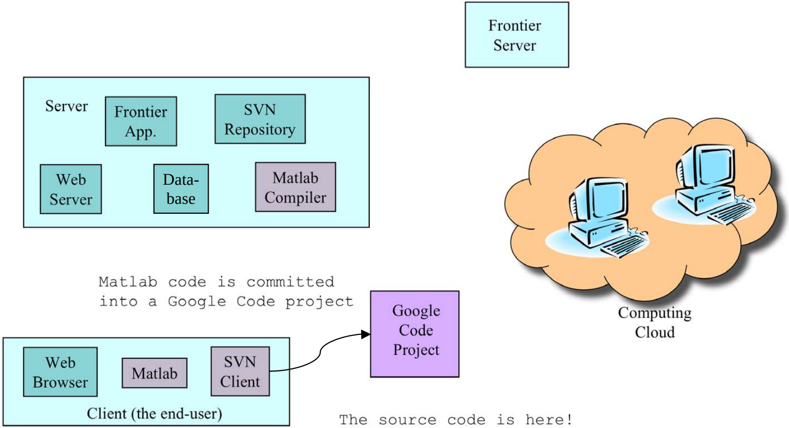
Client (the end-user)

Computing Cloud

# Running Custom Code

- We will allow any user to run jobs that use the current official release of CML.
- However, this requires that the desired simulation be already supported by CML.
- Authorized "power" users might like the ability to run code that they have developed on their own.
    - A certain level of trust is required because the compute engines will be running native code.

# Updating the Code



Frontier Server

Server

Frontier App.

SVN Repository

Web Server

Data-base

Matlab Compiler

Computing Cloud

Matlab code is committed into a Google Code project

Google Code Project

Web Browser

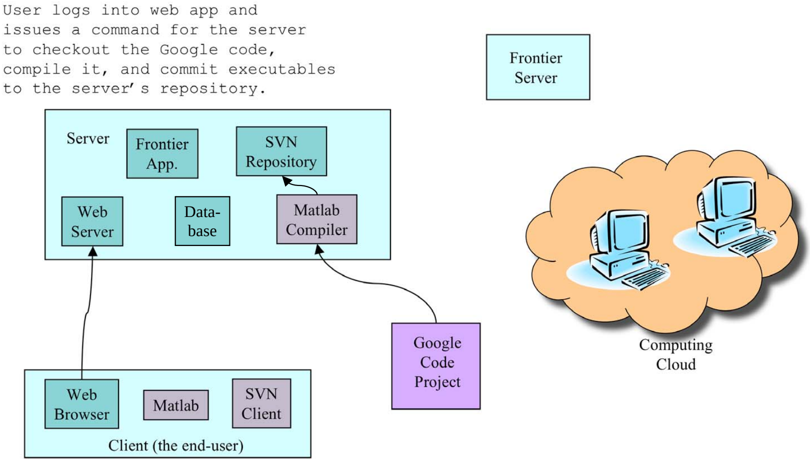
Matlab

SVN Client

Client (the end-user)

The source code is here!

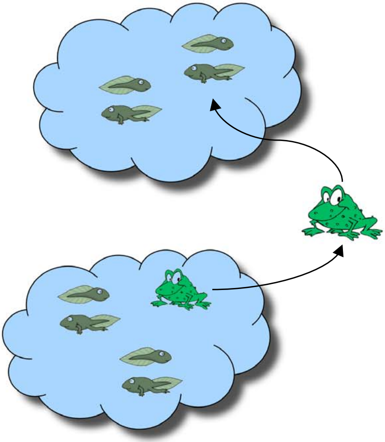
# Updating the Code


User logs into web app and issues a command for the server to checkout the Google code, compile it, and commit executables to the server' s repository.

# Genetic Algorithms

- Besides running simulations, the grid can be used for other compute-intensive tasks.
- The grid is well suited for evolutionary computing.
- In a genetic algorithm, a population of individuals evolves through breeding and mutation.
- Requires:
  - Genes appropriate to the application.
  - A fitness function.
  - Rules for breeding, mutating, and selection.

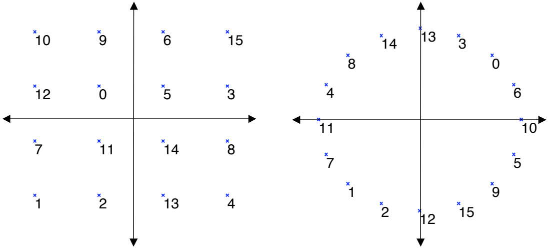
# Parallelizing Genetic Algorithms



- The evolution process can be accelerated when running on the grid.
- Each node can evolve its own gene pool.
- Periodically, the best individual among the pools is cloned and *immigrates* to other pools.

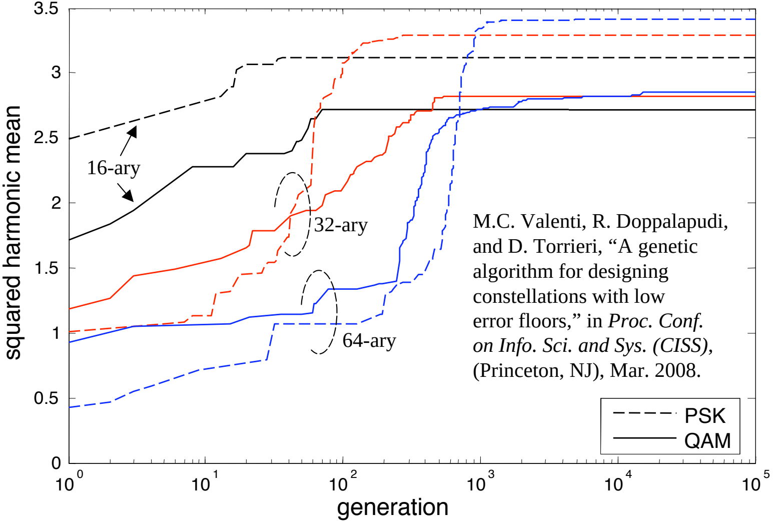
# Optimization of Signal Constellations

- Goal is to optimize the symbol labeling.



- Optimization criteria: Maximize the harmonic mean of the squared-Euclidian distances between signals whose labels differ in just one bit position.
    - Asympotically optimal for BICM-ID systems (Huang-Ritcey, 2005).

# Results of Optimization

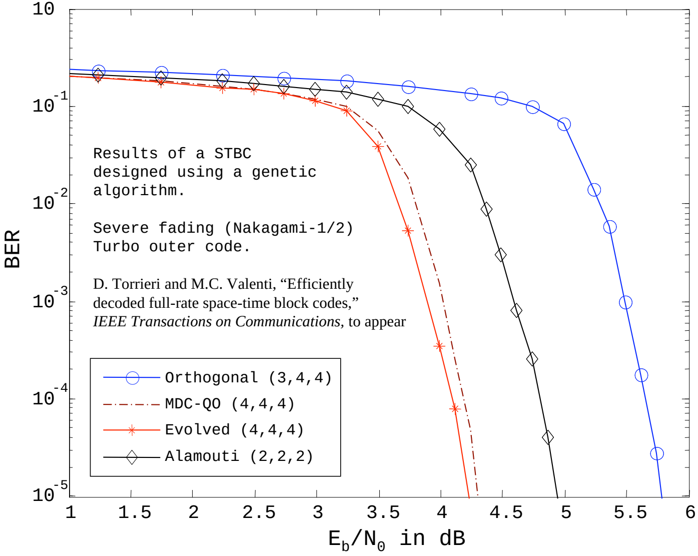


M.C. Valenti, R. Doppalapudi, and D. Torrieri, “A genetic algorithm for designing constellations with low error floors,” in *Proc. Conf. on Info. Sci. and Sys. (CISS)*, (Princeton, NJ), Mar. 2008.

# Optimization of STBC

- Space-time block codes enable communications with multiple transmit antennas.
- Desirable properties:
  - *Full-diversity:* To mitigate fading.
  - *Orthogonality:* Permits low-complexity "decoupled" decoding.
  - *Full-rate:* For spectral efficiency.
- For more than 2-TX antennas and a complex signal constellation, it is not possible to simultaneously satisfy all the above properties.
- Goal of evolved STBC's:
  - Force a full-rate code to be nearly orthogonal.
  - Use decoupled decoding despite resulting self-interference.

# Performance of Evolved STBC



Results of a STBC designed using a genetic algorithm.

Severe fading (Nakagami-1/2)
Turbo outer code.

D. Torrieri and M.C. Valenti, “Efficiently decoded full-rate space-time block codes,” *IEEE Transactions on Communications*, to appear

# Conclusions

- Computing is a commodity.
    - Utility computing = cheap and reliable.
    - Community computing = free but less reliable.
- By aggregating donated CPU cycles and providing a web-interface, we are developing a computing resource for the communication theory community.
- The computing grid is appropriate for:
    - Parallel Monte Carlo simulation.
    - Evolutionary computing.
- The resource is still under development, so please stay tuned!
    - Goal is to be online during the summer of 2010.

Thank you