This Thursday

• Do you like extra credit?

if (true)
  come to class Thursday with your computer
else
  do it anyway; we’ll talk about LIMITS of software

Project Final Presentations

• December 11, 10AM-11:15AM
• CS 150 (in the CS building)
• Think of this as a science fair.
• Each team will get an easel. Bring a poster or printed slides. And laptop for demo.
• Describe and discuss the solution, and demo the implementation.
• Will see (at least) 2 separate judges.
• Chance to see other projects too!

Today

Let’s talk about how to automate making high-quality software
Privacy and Reliability in an Untrusted Cloud
A private and secure cloud

Distributing computation onto untrusted machines.
Today’s focus on privacy: sTile

sTile
A technique for privately solving computationally-intensive problems (3-SAT) on untrusted computers.
Our approach: intelligent distribution

Obstacle: Private computation is hard and inefficient [Childs 2005; Gentry 2009].

Solution:
1. Divide computation into elemental subcomputations.
2. Distribute subcomputations onto network.
Computing with tiles

Input:

Program:

Computation: Copies of the program tiles self-attach to the input.
Addition with tiles

adding program
Addition with tiles

adding program

Encode input to add 10 ($= 1010_2$) and 11 ($= 1011_2$)
Addition with tiles

Adding program

Add the two least significant bits
Addition with tiles

adding program

Add the rest of the bits, one at a time: 10 + 11 = 21 (= 10101₂)

```
0 1 0 1 0 0
0 1 0 1 0 0
1 0 1 0 1 1
0 1 0 1 1 1
0 1 0 1 1 1
```
Addition with tiles

adding program

Suppose computers deployed tiles
Addition with tiles

Even if some were compromised, they couldn’t learn private data
3-SAT with tiles [Winfree 1998]

Addition [TCS’07]

3-SAT [Nat.Comp.’12]
sTile intuition: computers simulate tiles

1. set up the 3-SAT seed
sTile intuition: computers simulate tiles

2 the seed self-replicates

discovery algorithm
sTile intuition: computers simulate tiles

3 tiles recruit neighbors

secure multi-party computation [Yao 1986]
sTile intuition: computers simulate tiles

tiles recruit neighbors

secure multi-party computation [Yao 1986]
sTile intuition: computers simulate tiles

tiles recruit neighbors

secure multi-party computation [Yao 1986]
sTile intuition: computers simulate tiles

report solution to the client
Evaluation plan

- Formally prove privacy
- Empirically demonstrate robustness to network delay
- Empirically demonstrate scalability
Probability of reconstructing a 20-, 38-, and 56-bit input

\[
\Pr[\text{reconstruct}] = \frac{1}{e} 
\]

where

\[
e \approx 0.63
\]

When the input size is:

- 20 bits:
  - \(\approx 10^{-25}\)
- 38 bits:
  - \(\approx 10^{-35}\)
- 56 bits:
  - \(\approx 10^{-50}\)
sTile provides highly-probable privacy

**Threat model:**
A Byzantine fraction of the cloud attempts to reconstruct private data.

**sTile guarantee:**
\[
P_{\text{compromise}}(c, n, s) = 1 - (1 - c^n)^s
\]
where:
- \(c\) — compromised fraction
- \(n\) — bits in input
- \(s\) — number of seeds

**TeraGrid example**
Controlling \(\frac{1}{8}\) of TeraGrid’s 100,000 machines yields a probability of \(10^{-10}\) of data compromise of a 17-variable formula.
Experimental Setup

- **Mahjong**: sTile implementation framework
  - Java, 3K LoC, builds on Prism-MW [Malek et al. 2005]
  - Input: NP-c problem instance $P$
  - Output: Distributed software system to solve $P$
  - Download: [http://www.cs.umass.edu/~brun/Mahjong](http://www.cs.umass.edu/~brun/Mahjong)
Experimental Setup

- **Mahjong: sTile implementation framework**
  - Java, 3K LoC, builds on Prism-MW [Malek et al. 2005]
  - Input: NP-c problem instance $P$
  - Output: Distributed software system to solve $P$
  - Download: [http://www.cs.umass.edu/~brun/Mahjong](http://www.cs.umass.edu/~brun/Mahjong)

- **Networks**
  - 11-node private cluster (P4 1.5GHz, 512MiB, WinXP/2000)
  - 186-node USC HPCC cluster [High Performance Computing and Communications] (P4 Xeon 3GHz, Linux)
  - 100-node PlanetLab [Peterson et al. 2003]
    (global, varying speeds and resources)
Network Delay

Communication is \(~100\text{–}1000\) times faster in a CPU than on a network.
Communication is $\sim 100-1000$ times faster in a CPU than on a network.  

But latency is not throughput!
# Robustness to Network Delay

<table>
<thead>
<tr>
<th>Problem</th>
<th># of Nodes</th>
<th>Network Delay</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>11</td>
<td>Private Cluster</td>
<td>20.1 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HPCC</td>
<td>19.3 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PlanetLab</td>
<td>18.5 sec.</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>11</td>
<td>Private Cluster</td>
<td>41.6 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HPCC</td>
<td>41.2 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PlanetLab</td>
<td>43.9 min.</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>1,000,000</td>
<td>0ms</td>
<td>65 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10ms</td>
<td>57 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100ms</td>
<td>64 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500ms</td>
<td>60 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaussian</td>
<td>68 min.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distance-based</td>
<td>59 min.</td>
</tr>
</tbody>
</table>
## Robustness to Network Delay

<table>
<thead>
<tr>
<th>Problem</th>
<th># of Nodes</th>
<th>Network Delay</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mahjong</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>11</td>
<td>Private Cluster</td>
<td>20.1 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HPCC</td>
<td>19.3 sec.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PlanetLab</td>
<td>18.5 sec.</td>
</tr>
</tbody>
</table>

Network latency does not affect system throughput

<table>
<thead>
<tr>
<th>Distance</th>
<th># of Nodes</th>
<th>0ms</th>
<th>10ms</th>
<th>100ms</th>
<th>500ms</th>
<th>Gaussian</th>
<th>Distance-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000,000</td>
<td></td>
<td>65 min.</td>
<td>57 min.</td>
<td>64 min.</td>
<td>60 min.</td>
<td>68 min.</td>
<td>59 min.</td>
</tr>
</tbody>
</table>
# Scalability: Speed $\propto$ Network Size

<table>
<thead>
<tr>
<th>Network &amp; Problem</th>
<th># of Nodes</th>
<th>Execution Time</th>
<th>Speed-up Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Cluster</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>5</td>
<td>43 sec.</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{B}$</td>
<td>10</td>
<td>23 sec.</td>
<td>1.9</td>
</tr>
<tr>
<td>HPCC</td>
<td>93</td>
<td>220 min.</td>
<td>1.9</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>186</td>
<td>116 min.</td>
<td></td>
</tr>
<tr>
<td>PlanetLab</td>
<td>50</td>
<td>9.2 min.</td>
<td>1.9</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>100</td>
<td>4.8 min.</td>
<td></td>
</tr>
<tr>
<td>Simjong</td>
<td>125,000</td>
<td>8.7 hours</td>
<td>1.9</td>
</tr>
<tr>
<td>$\mathcal{E}$</td>
<td>250,000</td>
<td>4.5 hours</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>2.1 hours</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>1,000,000</td>
<td>64 min.</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Scalability: Speed \( \propto \) Network Size

<table>
<thead>
<tr>
<th>Network &amp; Problem</th>
<th># of Nodes</th>
<th>Execution Time</th>
<th>Speed-up Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Cluster A</td>
<td>5</td>
<td>43 sec.</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>23 sec.</td>
<td></td>
</tr>
<tr>
<td>HPCC</td>
<td>93</td>
<td>220 min.</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>186</td>
<td>116 min.</td>
<td></td>
</tr>
<tr>
<td>PlanetLab</td>
<td>50</td>
<td>9.2 min.</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>4.8 min.</td>
<td></td>
</tr>
<tr>
<td>Simjong</td>
<td>125,000</td>
<td>8.7 hours</td>
<td>1.9</td>
</tr>
<tr>
<td></td>
<td>250,000</td>
<td>4.5 hours</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500,000</td>
<td>2.1 hours</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>1,000,000</td>
<td>64 min.</td>
<td>2.0</td>
</tr>
</tbody>
</table>

System speed scales almost linearly with network size
Related Work

- Private computation in quantum computing through entanglement [Childs 2005]
- Homomorphic encryption for private computation [Gentry 2009]
- Plethora of non-private distributed computation work [BOINC 2009; Korpela et al. 1996; Larson et al. 2002; Rosetta@home; Dean and Ghemawat 2004; Chakravarti and Baumgartner 2004]
- ...and fault-tolerant computation work [Sarmenta 2002; Bondavalli et al. 1993, 2002; Felber and Schiper 2001; Koren and Krishna 2007; Hwang and Kesselman 2003]
- ...and private storage and access [Ateniese et al. 2006; Wang et al. 2011; Yang et al. 2011; Yu et al. 2010]
Contributions

sTile

- Distribution can result in privacy
- A bound on the cost of privacy


http://dx.doi.org/10.1109/TDSC.2013.13
Contributions

sTile
- Distribution can result in privacy
- A bound on the cost of privacy

How do I compute a function using Byzantine machines?
How do I send you a message over a noisy channel?
Environment model

A pool of network nodes

- Some nodes are Byzantine
- Byzantine node identity and rate are unknown
- Nodes may join, leave, fail, and become reliable

Smart redundancy: maximize task reliability for a given resource cost
Applicable to problems with many independent subtasks that can be executed out of order.

### Example
- MapReduce / Hadoop [Dean and Ghemawat 2004]
- Globus Grid Toolkit [Foster et al. 2001]
- BOINC [Korpela et al. 1996]
- reCAPTCHA [von Ahn et al. 2008]
- ESP Game [von Ahn and Dabbish 2004]
- FoldIt [Baker 2009]
- software verification [Schiller and Ernst 2010]
- AutoMan [Barowy et al. 2012]
Applicable to problems with many independent subtasks that can be executed out of order.

Example
- MapReduce / Hadoop [Dean and Ghemawat 2004]
- Globus Grid Toolkit [Foster et al. 2001]
- BOINC [Korpela et al. 1996]

Crowdsourcing applications too
- reCAPTCHA [von Ahn et al. 2008]
- ESP Game [von Ahn and Dabbish 2004]
- FoldIt [Baker 2009]
- software verification [Schiller and Ernst 2010]
- AutoMan [Barowy et al. 2012]
Voting redundancy

Assume (for now) we know average node reliability

If we ask 3 nodes, the system reliability will be:

\[ 1 - (0.3)^3 = 0.8419 \]

19 nodes have to vote to get 0.97 reliability:

\[ 1 - \sum_{i=1}^{19} \left( \frac{19!}{i!(19-i)!} \right) (0.3)^i (0.7)^{19-i} \approx 0.97 \]
Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7  desired system reliability: 0.97
Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7  desired system reliability: 0.97

If we ask 3 nodes, the system reliability will be:

\[ 1 - 0.3^3 - 3(0.3^2)0.7 \approx 0.84 \]
Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7  
desired system reliability: 0.97

- If we ask 3 nodes, the system reliability will be:
  
  \[1 - 0.3^3 - 3 \left(0.3^2\right) 0.7 \approx 0.84\]

- 19 nodes have to vote to get 0.97 reliability:
  
  \[1 - \sum_{i=10}^{19} \binom{19}{i} 0.3^i 0.7^{19-i} \approx 0.97\]
Smart redundancy

1. Distribute enough independent jobs to, in the best case, achieve desired reliability.
2. Compute reliability based on results.
3. Check if desired reliability achieved?
   - Yes: Solution
   - No: Repeat steps 1 and 2.

Main idea: Only deploy jobs if you need them.
## Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.70</td>
</tr>
</tbody>
</table>
# Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.70</td>
</tr>
<tr>
<td>2</td>
<td>(\frac{(0.7^2)}{(0.7^2)+(0.3^2)}) ≈ 0.84</td>
</tr>
</tbody>
</table>
## Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>( \frac{(0.7^2)}{(0.7^2)+(0.3^2)} ) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>( \frac{(0.7^3)}{(0.7^3)+(0.3^3)} ) \approx 0.93</td>
</tr>
</tbody>
</table>
## Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
</tbody>
</table>
| 2 0     | \[
\frac{(0.7^2)}{(0.7^2) + (0.3^2)} \approx 0.84
\] |
| 3 0     | \[
\frac{(0.7^3)}{(0.7^3) + (0.3^3)} \approx 0.93
\] |
| 3 1     | \[
\frac{3(0.7^3)0.3}{3(0.7^3)0.3 + 3(0.3^3)0.7} \approx 0.84
\] |
### Smart redundancy example execution

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>answers</td>
<td></td>
<td>reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.70</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.93</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.93</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

distribute enough independent jobs to, in the best case, achieve desired reliability

compute reliability based on results

desired reliability achieved?

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>(\frac{(0.7^2)}{(0.7^2)+(0.3^2)}) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>(\frac{(0.7^3)}{(0.7^3)+(0.3^3)}) \approx 0.93</td>
</tr>
<tr>
<td>3 1</td>
<td>(\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}) \approx 0.84</td>
</tr>
<tr>
<td>4 0</td>
<td>(\frac{(0.7^4)}{(0.7^4)+(0.3^4)}) \approx 0.97</td>
</tr>
<tr>
<td>4 1</td>
<td>(\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}) \approx 0.93</td>
</tr>
<tr>
<td>5 1</td>
<td>(\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}) \approx 0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>(\frac{(0.7^2)}{(0.7^2) + (0.3^2)}) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>(\frac{(0.7^3)}{(0.7^3) + (0.3^3)}) \approx 0.93</td>
</tr>
<tr>
<td>3 1</td>
<td>(\frac{3(0.7^3)0.3}{3(0.7^3)0.3 + 3(0.3^3)0.7}) \approx 0.84</td>
</tr>
<tr>
<td>4 0</td>
<td>(\frac{(0.7^4)}{(0.7^4) + (0.3^4)}) \approx 0.97</td>
</tr>
<tr>
<td>4 1</td>
<td>(\frac{4(0.7^4)0.3}{4(0.7^4)0.3 + 4(0.3^4)0.7}) \approx 0.93</td>
</tr>
<tr>
<td>5 1</td>
<td>(\frac{5(0.7^5)0.3}{5(0.7^5)0.3 + 3(0.3^5)0.7}) \approx 0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

distribute 4 jobs

compute reliability based on 3-1 split

desired reliability achieved?

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>( \frac{0.7^2}{0.7^2 + 0.3^2} ) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>( \frac{0.7^3}{0.7^3 + 0.3^3} ) \approx 0.93</td>
</tr>
<tr>
<td>3 1</td>
<td>( \frac{3 \cdot 0.7^3 \cdot 0.3}{3 \cdot (0.7^3 \cdot 0.3) + 3 \cdot (0.3^3) \cdot 0.7} ) \approx 0.84</td>
</tr>
<tr>
<td>4 0</td>
<td>( \frac{0.7^4}{0.7^4 + 0.3^4} ) \approx 0.97</td>
</tr>
<tr>
<td>4 1</td>
<td>( \frac{4 \cdot 0.7^4 \cdot 0.3}{4 \cdot (0.7^4 \cdot 0.3) + 4 \cdot (0.3^4) \cdot 0.7} ) \approx 0.93</td>
</tr>
<tr>
<td>5 1</td>
<td>( \frac{5 \cdot 0.7^5 \cdot 0.3}{5 \cdot (0.7^5 \cdot 0.3) + 3 \cdot (0.3^5) \cdot 0.7} ) \approx 0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

- Distribute enough independent jobs to, in the best case, achieve desired reliability.
- Compute reliability based on results.
- Desired reliability achieved?
- Yes → Solution
- No → Repeat the process.

<table>
<thead>
<tr>
<th>Answers</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>(\frac{0.7^2}{(0.7^2) + (0.3^2)}) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>(\frac{0.7^3}{(0.7^3) + (0.3^3)}) \approx 0.93</td>
</tr>
<tr>
<td>3 1</td>
<td>(\frac{3(0.7^3)0.3}{3(0.7^3)0.3 + 3(0.3^3)0.7}) \approx 0.84</td>
</tr>
<tr>
<td>4 0</td>
<td>(\frac{0.7^4}{(0.7^4) + (0.3^4)}) \approx 0.97</td>
</tr>
<tr>
<td>4 1</td>
<td>(\frac{4(0.7^4)0.3}{4(0.7^4)0.3 + 4(0.3^4)0.7}) \approx 0.93</td>
</tr>
<tr>
<td>5 1</td>
<td>(\frac{5(0.7^5)0.3}{5(0.7^5)0.3 + 3(0.3^5)0.7}) \approx 0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>(\frac{(0.7^2)}{(0.7^2)+(0.3^2)}) \approx 0.84</td>
</tr>
<tr>
<td>3 0</td>
<td>(\frac{(0.7^3)}{(0.7^3)+(0.3^3)}) \approx 0.93</td>
</tr>
<tr>
<td>3 1</td>
<td>(\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}) \approx 0.84</td>
</tr>
<tr>
<td>4 0</td>
<td>(\frac{(0.7^4)}{(0.7^4)+(0.3^4)}) \approx 0.97</td>
</tr>
<tr>
<td>4 1</td>
<td>(\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}) \approx 0.93</td>
</tr>
<tr>
<td>5 1</td>
<td>(\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}) \approx 0.97</td>
</tr>
</tbody>
</table>
Smart redundancy example execution

<table>
<thead>
<tr>
<th>answers</th>
<th>reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0.70</td>
</tr>
<tr>
<td>2 0</td>
<td>(\frac{0.7^2}{0.7^2 + 0.3^2}) (\approx 0.84)</td>
</tr>
<tr>
<td>3 0</td>
<td>(\frac{0.7^3}{0.7^3 + 0.3^3}) (\approx 0.93)</td>
</tr>
<tr>
<td>3 1</td>
<td>(\frac{3(0.7^3)0.3}{3(0.7^3)0.3 + 3(0.3^3)0.7}) (\approx 0.84)</td>
</tr>
<tr>
<td>4 0</td>
<td>(\frac{0.7^4}{0.7^4 + 0.3^4}) (\approx 0.97)</td>
</tr>
<tr>
<td>4 1</td>
<td>(\frac{4(0.7^4)0.3}{4(0.7^4)0.3 + 4(0.3^4)0.7}) (\approx 0.93)</td>
</tr>
<tr>
<td>5 1</td>
<td>(\frac{5(0.7^5)0.3}{5(0.7^5)0.3 + 3(0.3^5)0.7}) (\approx 0.97)</td>
</tr>
</tbody>
</table>

compute reliability based on 5-1 split

desired reliability achieved?

yes

solution

no

distribute 2 more jobs
smart redundancy
(1) assumes best case and asks the minimum number of nodes
(2) asks more after learning how reality differs from best case.
How many jobs to distribute?

<table>
<thead>
<tr>
<th>room 1</th>
<th>Flip a 70% / 30% coin 4 times get 4 heads and 0 tails.</th>
</tr>
</thead>
<tbody>
<tr>
<td>room 2</td>
<td>Flip a 70% / 30% coin 1004 times get 504 heads and 500 tails.</td>
</tr>
</tbody>
</table>
How many jobs to distribute?

**room 1**
Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

**room 2**
Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

\[
\frac{\binom{1004}{504} \cdot 0.7^{504} \cdot 0.3^{500}}{\binom{1004}{504} \cdot 0.7^{504} \cdot 0.3^{500} + \binom{1004}{500} \cdot 0.3^{504} \cdot 0.7^{500}}
\]
How many jobs to distribute?

**room 1**
Flip a 70% / 30% coin 4 times get 4 heads and 0 tails.

**room 2**
Flip a 70% / 30% coin 1004 times get 504 heads and 500 tails.

\[
\frac{\binom{1004}{504}(0.7504)^{0.3500}}{\binom{1004}{504}(0.7504)^{0.3500} + \binom{1004}{500}(0.3504)^{0.7500}}
\]
How many jobs to distribute?

room 1
Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2
Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

\[
\frac{\binom{1004}{504}(0.7504)^{0.3500}}{\binom{1004}{504}(0.7504)^{0.3500} + \binom{1004}{500}(0.3504)^{0.7500}}
\]
How many jobs to distribute?

**room 1**
Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

**room 2**
Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

\[
\frac{\begin{pmatrix} 1004 \\ 504 \end{pmatrix} (0.7504)^{504} (0.3500)^{500}}{\begin{pmatrix} 1004 \\ 504 \end{pmatrix} (0.7504)^{500} (0.3500)^{504} + \begin{pmatrix} 1004 \\ 504 \end{pmatrix} (0.7504)^{500} (0.3500)^{500}}
\]
How many jobs to distribute?

**Room 1**
Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

\[
\frac{0.7^4}{0.7^4 + 0.3^4} = \frac{1004}{504}(0.7504)0.3500
\]

**Room 2**
Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

\[
\frac{1004}{504}(0.7504)0.3500 + \frac{1004}{500}(0.3504)0.7500
\]
How many jobs to distribute?

room 1
Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

\[
\frac{0.7^4}{0.7^4 + 0.3^4} = \frac{1004 \cdot 0.7504 \cdot 0.3500}{1004 \cdot 0.7504 \cdot 0.3500 + 1004 \cdot 0.3504 \cdot 0.7500}
\]

Bayes theorem implies that given an a-b split of answers, only the difference affects the reliability.

room 2
Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.
Inject redundancy only when it is needed

node reliability:

cost factor:

system reliability:
Smart always outperforms voting redundancy

Theoretical results

![Graph showing the comparison between Smart (SR) and Voting Redundancy (VR) in terms of cost factor and reliability. The graph illustrates that Smart consistently outperforms Voting Redundancy across various reliability levels.](image-url)
Simulation analysis confirms theoretical predictions

Simulated 1,000,000 task executions on 10,000 nodes using the XDEVS simulator [Edwards 2010]
Empirical analysis confirms theoretical predictions

Deployed a SAT solver using BOINC [Anderson 2004] on PlanetLab [Peterson et al. 2003]
Response time cost

Iterating increases individual task response time
Related work

other redundancy techniques

- self-configuring optimistic programming [Bondavalli et al. 2002]
- credibility-based fault tolerance [Sarmenta 2002]
- checkpointing [Priya et al. 2007]
- crowdsourcing [Barowy et al. 2012]
- Byzantine faults in service-based computing (ZZ [Wood et al. 2011])

complementary

- primary backup [Budhiraja et al. 1993]
- active replication [Schneider 1990]
- developer-defined fault detection [Hwang and Kesselman 2003]
Contributions and Future Projects

smart redundancy: using resources optimally to boost reliability

What’s next?

- Channels with more bandwidth than 1 bit
- Using history to improve resource use (non-Byzantine)
- Crowdsourcing
Contributions and Future Projects

What’s next?

- Channels with more bandwidth than 1 bit
- Using history to improve resource use (non-Byzantine)
- Crowdsourcing

For more, see “Smart redundancy for distributed computation” by Y. Brun et al. In the 31st International Conference on Distributed Computing Systems (ICDCS), 665-676, 2011. http://dx.doi.org/10.1109/ICDCS.2011.25


Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky. SETI@home — massively distributed computing for SETI. *IEEE MultiMedia*, 3(1):78–83, 1996. ISSN 1070-986X.

Stefan M. Larson, Christopher D. Snow, Michael R. Shirts, and Vijay S. Pande. *Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology*. Horizon Press, 2002.


