Smart Redundancy for Distributed Computation

Yuriy Brun
George Edwards
Jae young Bang
Nenad Medvidović

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]

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]

Voting redundancy

Assume (for now) we know average node reliability

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: $1 - 0.3^3 - 3\left(0.3^2\right)0.7 \approx 0.84$

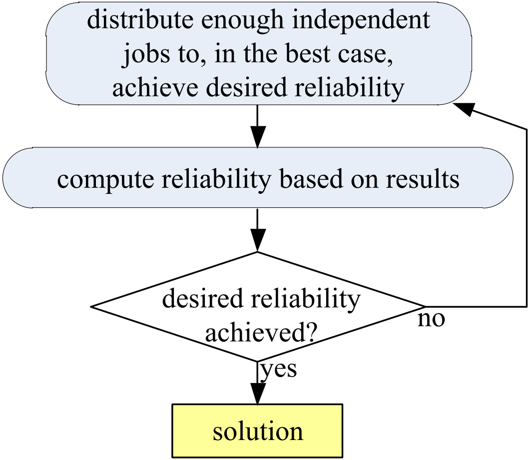
Voting redundancy

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



main idea: only deploy jobs if you need them

Smart redundancy example execution

answers		reliability
1	0	0.70

Smart redundancy example execution

answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84

Smart redundancy example execution

answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93

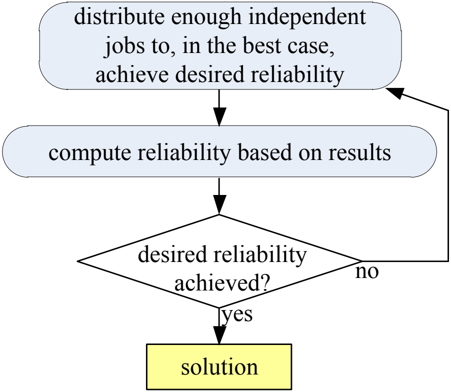
Smart redundancy example execution

answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84

Smart redundancy example execution

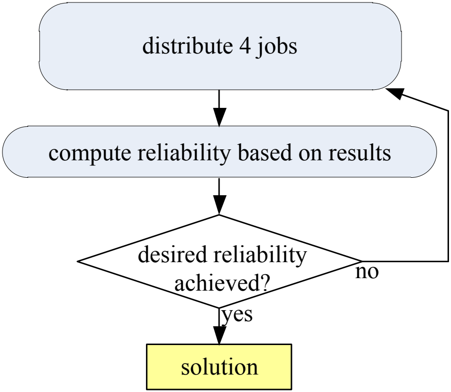
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



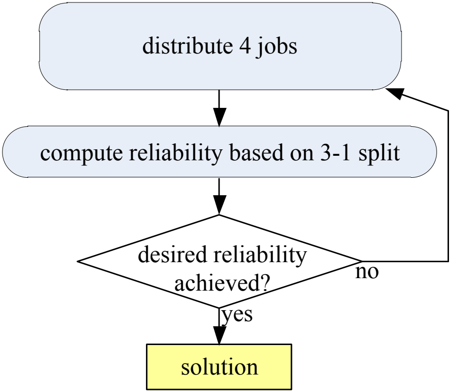
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



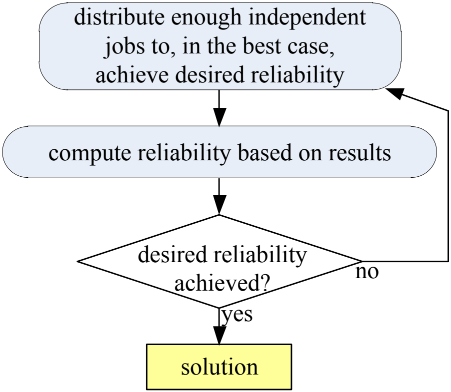
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



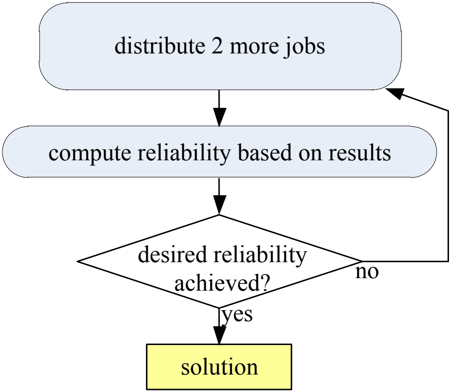
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



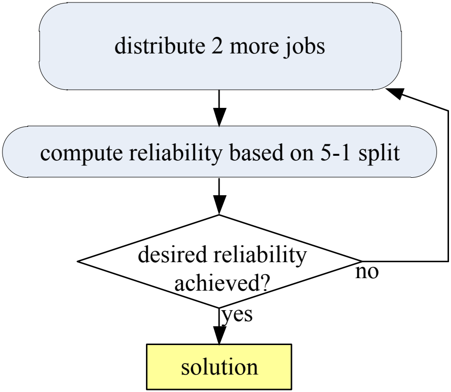
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

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?

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

How many jobs to distribute?

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

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.

How many jobs to distribute?

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

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.

$$\frac{\binom{1004}{504}\left(0.7^{504}\right)0.3^{500}}{\binom{1004}{504}\left(0.7^{504}\right)0.3^{500}+\binom{1004}{500}\left(0.3^{504}\right)0.7^{500}}$$

How many jobs to distribute?

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

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.

$$\frac{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)0.3^{500}}{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)0.3^{500}+\cancel{\binom{1004}{500}}\left(0.3^{504}\right)0.7^{500}}$$

How many jobs to distribute?

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

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.

$$\frac{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}\left(0.3^{\cancel{504}4}\right)0.7^{500}}$$

How many jobs to distribute?

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

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.

$$\frac{\cancel{\binom{1004}{504}}\left(0.7^{\cancel{504}4}\right)\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}\left(0.7^{\cancel{504}4}\right)\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}\left(0.3^{\cancel{504}4}\right)\cancel{0.7^{500}}}$$

How many jobs to distribute?

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

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.

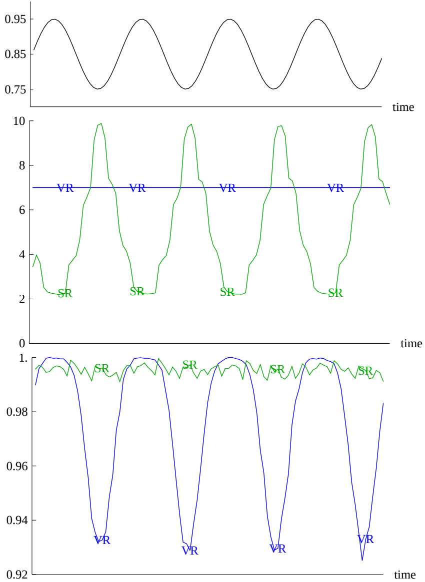
$$\frac{(0.7^4)}{(0.7^4)+(0.3^4)} = \frac{\cancel{\binom{1004}{504}}(0.7^{\cancel{5}04})\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}(0.7^{\cancel{5}04})\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}(0.3^{\cancel{5}04})\cancel{0.7^{500}}}$$

Inject redundancy only when it is needed

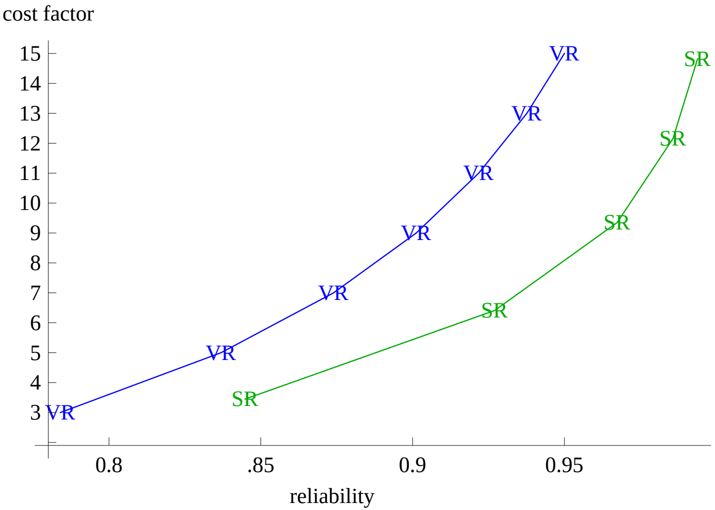
node reliability:

cost factor:

system reliability:

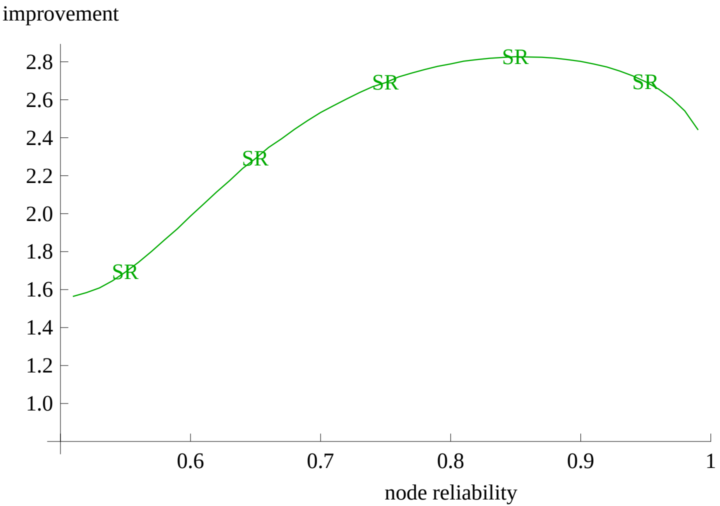


Smart always outperforms voting redundancy



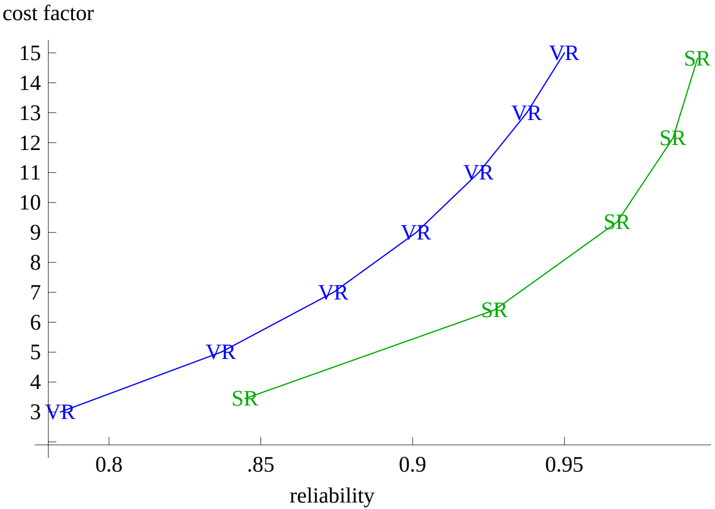
node reliability is 0.7

Reliable nodes lead to greater benefits



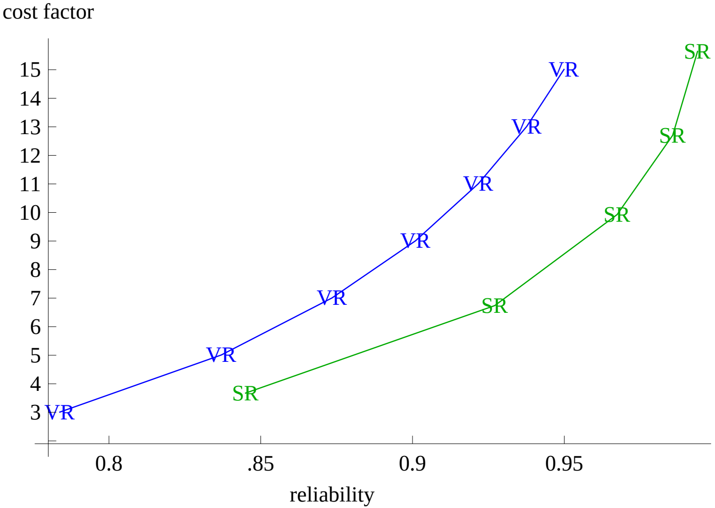
Smart always outperforms voting redundancy

Theoretical results



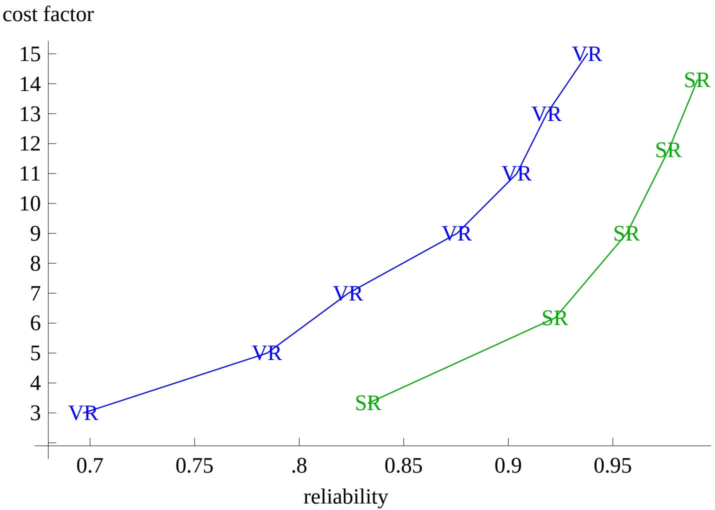
Simulation analysis confirms theoretical predictions

Simulated 1,000,000 task executions on 10,000 nodes using the XDEVS simulator [Edwards 2010]

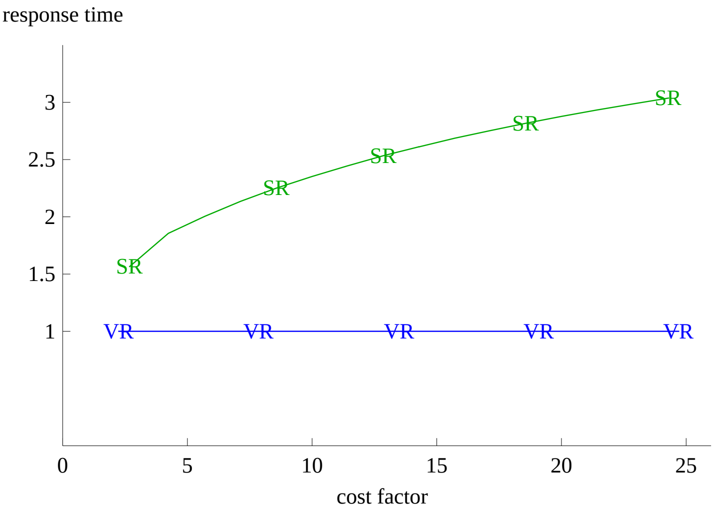


Empirical analysis confirms theoretical predictions

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

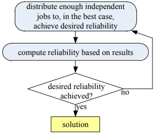
other redundancy techniques

- self-configuring optimistic programming [Bondavalli et al. 2002]
- credibility-based fault tolerance [Sarmenta 2002]
- checkpointing [Priya et al. 2007]

complementary

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

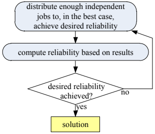
Contributions



smart redundancy: using resources optimally to boost reliability

Yuriy Brun: brun@cs.umass.edu

Contributions

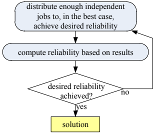


smart redundancy: using resources optimally to boost reliability

developed a simple, efficient, self-adaptive implementation

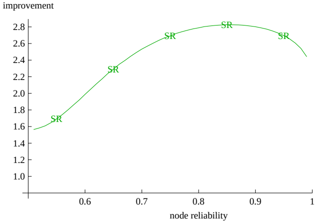
Yuriy Brun: brun@cs.umass.edu

Contributions



smart redundancy: using resources optimally to boost reliability

developed a simple, efficient, self-adaptive implementation



formally and empirically verified improvement

Yuriy Brun: brun@cs.umass.edu

David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID04)*, pages 4–10, Pittsburgh, PA, USA, 2004. ISBN 0-7695-2256-4. doi: http://dx.doi.org/10.1109/GRID.2004.14.

David Baker. Foldit. http://fold.it, 2009.

Andrea Bondavalli, Silvano Chiaradonna, Felicita Di Giandomenico, and Jie Xu. An adaptive approach to achieving hardware and software fault tolerance in a distributed computing environment. *Journal of Systems Architecture*, 47(9):763–781, 2002. ISSN 1383-7621. doi: 10.1016/S1383-7621(01)00029-7.

Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. The primary-backup approach. In *Distributed Systems*, pages 199–216. ACM Press/Addison-Wesley Publishing Co., 2 edition, 1993. ISBN 0-201-62427-3.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI04)*, San Francisco, CA, USA, December 2004.

George T. Edwards. *Automated Synthesis of Domain-Specific Model Interpreters*. PhD thesis, University of Southern California, Los Angeles, CA, USA, 2010.

Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001. ISSN 1094-3420. doi: 10.1177/109434200101500302.

Soonwook Hwang and Carl Kesselman. A flexible framework for fault tolerance in the grid. *Journal of Grid Computing*, 1(3): 251–272, September 2003. ISSN 1570-7873. doi: 10.1023/B:GRID.0000035187.54694.75.

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.

Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the Internet. *ACM SIGCOMM Computer Communication Review*, 33(1):59–64, 2003. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/774763.774772.

S. Baghavathi Priya, M. Prakash, and K. K. Dhawan. Fault tolerance-genetic algorithm for grid task scheduling using check point. In *Proceedings of the 6th International Conference on Grid and Cooperative Computing (GCC07)*, pages 676–680, 2007. ISBN 0-7695-2871-6. doi: 10.1109/GCC.2007.67.

Luis F. G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002. ISSN 0167-739X. doi: 10.1016/S0167-739X(01)00077-2.

Todd W. Schiller and Michael D. Ernst. Rethinking the economics of software engineering. In *Workshop on the Future of Software Engineering Research*, Santa Fe, NM, USA, November 2010. doi: 10.1145/1882362.1882429.

Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22:299–319, December 1990. ISSN 0360-0300. doi: 10.1145/98163.98167.

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI04)*, pages 319–326, Vienna, Austria, 2004. doi: 10.1145/985692.985733.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008. doi: 10.1126/science.1160379.