The ChessBrain Project

A Global Effort To Build The World's Largest Chess SuperComputer

Carlos Justiniano and Colin M. Frayn
Newbury Park, USA / Cambridge, UK

Published in the June 2003 issue of the
Journal of the International Computer Games Association (ICGA)
Vol. 26, No. 2, pp. 132-138.

ABSTRACT

This article examines a distributed computing project that uses the processing power of Internet-connected machines to construct a single chess-playing supercomputer.

1. INTRODUCTION

In the past decade, researchers began investigating how to link commodity-off-the-shelf (COTS) computers to create less expensive supercomputers. In the summer of 1994, Thomas Sterling and Don Becker had a problem they wanted to solve; they decided to use commonly available components to build a 16-computer cluster that functioned as a single machine. They named their machine Beowulf, and it quickly became a success within NASA, and the academic and research communities (Beowulf, 1994). The Beowulf project demonstrated the viability of using COTS computers to solve computational problems. Over time, Beowulf clusters were built using hundreds of machines.

The rapid growth of the Internet brought about several opportunities to extend the Beowulf concept to include machines distributed throughout the Internet. As computers became faster and cheaper, many spent nearly all of their idle time waiting for something to happen. The idea of salvaging unused CPU time became synonymous with distributed computing. This attracted more than academic attention when a data security company, RSA Security Inc. (http://www.rsasecurity.com/) began offering cash prizes for cracking encryption schemes.

In the spring of 1997, a group of individuals began preparing to compete in a challenge to solve RSA's 56-bit encryption algorithm. The group set out to develop network servers that could coordinate the remote machines. The task involved potentially testing 72 quadrillion encryption keys by distributing the work of testing to as many computers as possible. Given time, brute-force attempts can be used to crack encryption schemes. Hence, the value of any encryption scheme is directly proportionate to the time required to break it. At the time, the US government maintained that a 56-bit encryption key offered sufficient security for business. The traditional methods of solving this sort of problem involved using supercomputers. As summer approached the group gained exposure via the Internet and became formally known as distributed.net. By October, distributed.net aided by thousands of contributors discovered the correct key to solve the 56-bit challenge, which revealed RSA's blunt statement: "The unknown message is: It's time to move to a longer key length".

The distributed.net project proved the viability of massively distributed computations. In addition, distributed.net demonstrated that using the contributed machines of unknown individuals over a public network could provide viable alternatives to centralized supercomputers and clusters. The result gained immediate attention from technical publications, and on October 24th the New York Times (1997) published an article "Cracked Code Reveals Security Limits" on the achievement.

The most widely publicized distributed computing project began in 1998, when a group of researchers at the University of California in Berkeley launched the SETI@home project. Millions of people contributed the use of their machines to the SETI@home (The Search for Extraterrestrial Intelligence) project. Researcher David Anderson wrote that at the time the fastest supercomputer was the ASCI White built by IBM for the U.S. Department of Energy. The machine cost US $110 million, weighed 106 tons and offered a peak performance of 12.3 TFLOPS (Oram, 1998). Anderson wrote that SETI@home was faster than the ASCI White at less than 1% of the cost. The reason for the remarkable cost difference stemmed from distributing the cost of computation. Each remote contributor bares the burden of operating a compute node – including the cost of electricity and any necessary upgrades. If the SETI@home project was faced with buying the time on a supercomputer the project might not exist. Soon more distributed computing projects were formed. Each, seeking to tap the vast and growing seas of underutilized computing resources throughout the Internet.

For many people, having the opportunity to play a small role in a potentially significant achievement outweighs the personal costs of participation. Still, conventional wisdom might dictate that most personal computer users would demand to be paid for their contributions. After all, people who allow the use of their equipment have to cover the costs of electricity and in many cases, costly Internet connections. In a stroke of genus, the SETI@home project packaged their remote client software as a screensaver. Offering users a screensaver application had several key advantages.

1. The very thought of a screensaver seems innocent, familiar, and non-intrusive.

2. People already know that screensavers only start when they do not use their computer, and it does not interfere with their use of the machine.

3. Many people like to show off their screensavers (rather than simply turn off their monitors) and so, screensavers are good advertising tools. A great many people learned about SETI@home by seeing it run on a friend's computer. This added an automatic and visual component to the "word of mouth" marketing.

Today, SETI@home has had over three million participants. Other projects have followed in their footsteps. As we write, the Google Web Directory (2003) lists 87 distributed computing projects. With goals ranging from biology, mathematics, physics, and even the fight against bio terrorism, there seems to be a project for everyone. In recent news a Hong Kong based software company claimed that its distributed computing software, with the help of Internet based contributors, can predict the spread of the SARS virus (Lui, 2003).

Distributed computing technologies such as distributed computation, file sharing, and other P2P collaboration tools are poised to impact our social and technological evolution. The good news for researchers is that the masses are all too willing to participate in exciting and new research projects!

2. CHESSBRAIN

The ChessBrain project was created in the summer of 2001 to explore distributed computation and the game of Chess. Inspired by the pioneering work at Distributed.net and SETI@home, we set out to construct a massive chess-playing computer by using hundreds of Internet-connected machines.

During the first six months of the project, we focused exclusively on creating the two essential components of distributed computing software, the server and the client.

We created a custom server application, the SuperNode Server, to handle the creation and distribution of work to remote client machines across the Internet. The SuperNode itself consists of two major components, a network server application and a chess-playing component. The chess and server modules work independently and communicate with one another using operating system pipes. We decoupled the server application from the chess application to create a loose binding to enable the replacement of the core game component within the SuperNode machine.

The SuperNode server contains a BOT (software robot) called Shannon (in homage to Claude Shannon). Shannon communicates with Internet-based game servers and understands commands that allow it to log on and off of a game server and, accept, initiate, and resume games. Shannon relays game state information to the SuperNode server, which in turn communicates with the chess component to begin processing and work unit distribution.

Next, we worked on the remote client software, which we named the PeerNode. We chose the name "PeerNode" because, like the SuperNode, the PeerNode is actually a server application itself, albeit scaled down, and fully capable of communicating with other peers. At present, PeerNodes do not communicate with other PeerNodes; however, this powerful feature has far reaching possibilities in the future. Thus, the PeerNode is a client of the SuperNode, but a server in its own right. Software that behaves like a client and server is common among Peer-to-Peer applications. A good example is an Instant Messaging application, which communicate with a central server and is able to establish peer connections with one or more IM buddies. Another example is the file sharing applications, which allow users to traverse a network looking for files, which are ultimately transferred from a single machine to another machine via a Peer-to-Peer connection.

Each PeerNode client includes a communication component (scaled down server), which receives the work units and later transmits the results, and a chess-engine component, which completes assigned work units. We will explore the relationship with the SuperNode and PeerNodes in Section 3.

From a high-level view, the SuperNode works with hundreds of PeerNodes to play chess on a game server (see Figure 1). Individual members on a game server are able to play against ChessBrain, while many other members can observe the active game. ChessBrain can only play one game at a time, but does so using hundreds (and someday thousands) of machines.

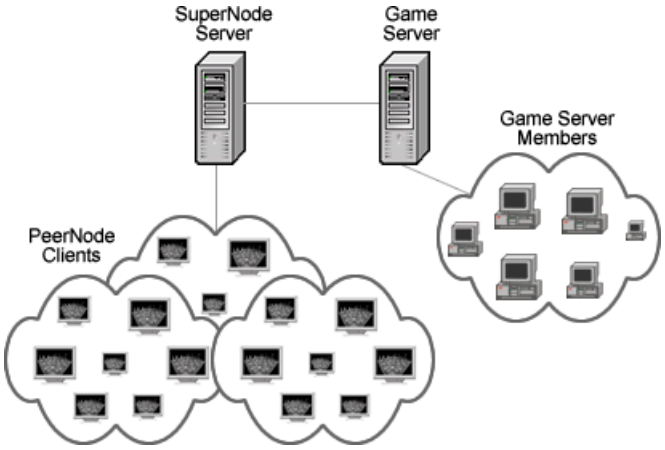

Figure 1: The Structure of ChessBrain.

Once the software was complete, we deployed two Pentium III 1 GHz machines to high-speed networks, one in southern California on a T1 connection, and one in northern California on T3 connection. Network hosting was generously provided by two long standing ChessBrain contributors, Walt Howard (HackerWhacker.com) and Gavin M. Roy (bteg.net).

After a few days of testing we released multiple versions of the PeerNode Client for use on machines running Microsoft Windows, Linux, Apple MacOSX and FreeBSD. Soon hundreds of people downloaded the software and enabled their machines to begin participating. The sudden surge of interest surprised us and caused a significant setback in our development efforts, as the project provided support to hundreds of people. Many had questions regarding the project, some wanted to help, and many offered useful feedback.

In October our efforts moved into high gear. We focused on preparing ChessBrain to play its first complete game of chess. On Tuesday, December 17th 2002 ChessBrain played its first distributed game of Chess. ChessBrain becomes the first distributed computation project to play games online using hundreds of computers throughout the Internet.

2.1 Project goals

ChessBrain was created with the initial goal of exploring distributed chess computation. Over time, other goals surfaced, introducing larger, more ambitious opportunities. We wanted ChessBrain to become a prototype for future distributed game projects. From a software development perspective, ChessBrain was constructed to allow its game-playing capabilities to be easily replaced. We realized that as the project gained exposure others would become interested in participating by offering their own ideas, and we wanted to ensure that the software infrastructure was geared to accept new game-playing components. In addition, the project's source code was structured to help make other distributed game projects possible. We would love to see a massively distributed Go project on the Internet someday.

Prior to starting the project we saw significant challenges in distributed game computation, and recognized that some regard such efforts as an exercise in futility. However, we also recognized that history is replete with examples where common belief has done little more than stifle innovation. Thirteenth century mystic poet Jelaluddin Rumi wrote: "Out beyond ideas of wrongdoing and rightdoing, there is a field. I'll meet you there." The field where we meet is the single largest man-made structure ever created, a global planetary network called the Internet. We are confident that challenges may be overcome through open collaboration. While developing ChessBrain a few major sub goals became clear, and can be characterized as Autonomous, Secure, Stable, Scalable, and Free.

2.1.1 Autonomous

Once the ChessBrain network could play, it would have to do so without human intervention. The goal was to see ChessBrain play on ICC-style game servers with an emphasis on the Free Internet Chess Server (http://www.freechess.org).

2.1.2 Secure, Stable, and Scalable

One of the most important challenges involved making ChessBrain secure to prevent malicious users from compromising the network and ultimately the accuracy of play. An early attempt involved the use of the Blowfish encryption algorithm. Blowfish proved not to be endian friendly, forcing us to consider an endian neutral algorithm. This led to the use of the Advanced Encryption Standard (AES-Rijndael, 200X). The ChessBrain SuperNode was developed as a multithreaded current connection server using techniques such as connection pooling and caching. This design makes the server scalable on multi-processor machines and portable to a number of platforms.

Perhaps one of the biggest issues facing distributed computing projects is how to manage effectively the network bandwidth. Bandwidth utilization can range from hundreds of dollars per month to several thousand for highly successful projects. To address this issue a strict limit was established to restrict the size of our data transmission. In addition we employed data compression using the patent free ZLib data compression algorithm (Zlib, 200X). The use of both compression and encryption renders the data unintelligible to network intrusion tools such as network diagnostic (sniffers) software.

2.1.3 Free

By free, we mean free. The ChessBrain project is both non-profit and noncommercial. The project is currently supported by hundreds of contributors who donate their time and resources. In order to help ensure ChessBrain's future we are preparing to release it as an open source project. We hope to see a great many more people take interest in exploring distributed computation using games. Once the project is freely and easily accessible we see no reason why it could not be adapted for use in machine-learning experiments involving a massive network of readily available machines.

3. HOW CHESSBRAIN WORKS

Chess offers tremendously difficult problems to solve, but lends itself conveniently to the concept of distributed computing. Human players, in order to analyze a position properly, must test many of the available moves in turn and ascertain which of these moves leads to the best possible end result. A computer uses a remarkably similar algorithm. The extension of this problem to parallel computing is evident: while one computer analyzes one move, another could equally well be analyzing the next, as the results are not interdependent. With additional computers all mutually connected this could continue until one computer analyzes the consequences of each of the possible moves from any given position. With more computers working on the problem the workload may be broken down still further and the search can proceed even more quickly. Each computer added to the task potentially makes this expanding network a fraction stronger.

This, in all its simplicity, is the principle behind the ChessBrain project. Though the theory seems simple, the practical implementation poses a myriad of problems. Not least of which is: how are many inter-linked computers coordinated so that the game tree is searched as efficiently and accurately as possible? This is addressed by introducing a central server to which all of the other machines are connected via the Internet. This central server, or SuperNode, acts as a director, scheduling and distributing the required jobs and sorting through the results when they are returned.

The SuperNode is the central brain of ChessBrain. It is fully capable of playing chess on its own when time is short, or when it finds itself in recognized positions. However, whenever the SuperNode reaches an unfamiliar position the many PeerNodes are brought to bear on the problem. The method by which this daunting workload is synchronized is of fundamental importance and the responsibility all rests on the central SuperNode code. ChessBrain uses an algorithm loosely based on the APHID method developed by Mark Brockington (1997) for asynchronous parallel game-tree search (see Figure 2). Initially, the SuperNode performs a shallow analysis of the unfamiliar position for a fixed time limit. This gives an estimation of the potentially strong moves, provides the data required for accurate 'best-first' move ordering and of course allows the SuperNode quickly to pick out shallow checkmate lines which need not be searched further.

Once this stage is complete, the SuperNode continues analyzing the game tree, splitting up and prioritizing the work to be done based on what it already knows. It performs repeated shallow passes through the upper levels of the game tree, down to two ply initially, and creates a prioritized list of all the leaf nodes which must be searched in order to complete an accurate evaluation. Each leaf node becomes a work unit, consisting of a board position and a depth to which that position should be analyzed by the assigned PeerNode. The SuperNode ensures that these work units are small enough to be completed quickly by a single computer. This is achieved using a simple cost-prediction scheme based on the previous computational complexity of the position to be searched if the prediction is known; otherwise it is achieved using an estimate based on the cost for the parent position and an approximation to the branching complexity at that node.

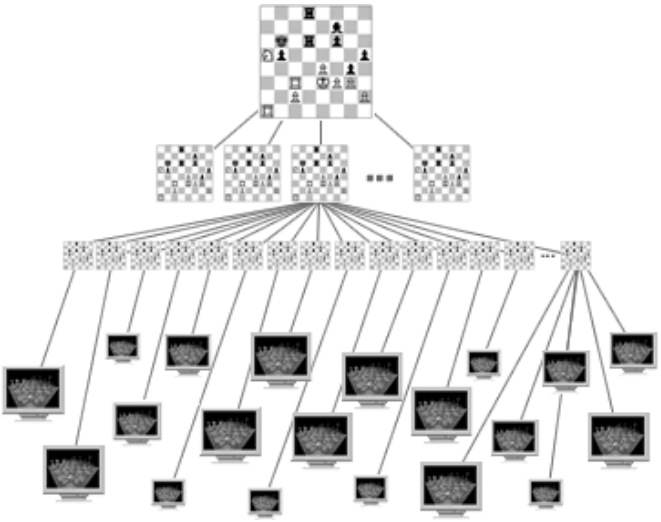

Figure 2: ChessBrain performing an asynchronous parallel game-tree research.

More complex jobs are assigned to the PeerNodes deemed most likely to return results in the shortest time; work units lying in the principal variation are prioritized above those further down the move list. All of the potential moves must be distributed initially, many of which will become obsolete. As work units are returned, the alpha-beta values for each node are updated, and daughter nodes are deleted if necessary, due to fail-high cuts. Deleted nodes are then removed from the prioritized work list. Moreover, the information is made available to the relevant PeerNodes to abort their now-redundant task and seek further instructions. If any individual work unit is deemed to have taken too long

to complete then it is automatically aborted and its reduced-depth score is returned. The SuperNode estimates whether a full-depth search was likely to have produced a score within the alpha-beta window. If so, then the node is further subdivided and re-distributed to the fastest PeerNodes available. If not, then the incomplete result is accepted without a full re-search.
Eventually, the SuperNode obtains an exact score of the root node, and the algorithm returns the suggested best move and the root score. If there is sufficient time left on the clock then the analysis continues to the next ply, retaining as much information from the previous iteration as possible and splitting up the new work units as the increased search depth makes their predicted search complexity too large to be handled by any one single machine in the network.

There are many safety precautions taken with this approach. Most importantly, the SuperNode sends out each work unit to multiple PeerNodes in order to facilitate error detection. This tactic also ensures that a result is returned as quickly as possible: the more computers allocated to each work unit, the less likely we are to encounter problems with PeerNodes burdened by other duties, or crippled by network lag. Naturally, some sort of balance must be struck between sending out as many work units as possible and ensuring that the results from those units are returned as quickly and as reliably as possible. This sort of balance can only be determined by extended periods of testing.

4. CONCLUSION

ChessBrain works - it now plays games of chess at around International Master level. However, this is changing at an exciting rate. With each round of testing, ChessBrain improves. At each stage we introduce improved methods for dividing up the computational workload, and for getting as much performance as possible out of our rapidly growing network of PeerNodes. On May 8th, 2003 ChessBrain used 531 machines while playing games of chess. We expect the total number of machines to be more than double within the next few months.

The ChessBrain project has over 800 hundred registered members from 37 different countries around the world. Many ChessBrain members contribute the use of one or more machines, with several members contributing over 20 machines. On average, 300 to 400 computers are active during any given game, and during the past few months the network has processed over 20 trillion positions. We recently completed testing ChessBrain on the Global Chess Servers at GamesParlor.com and watched ChessBrain play for over 20 hours without human intervention.

To continue to the raise public awareness for the project we are working to help stage a public exhibition involving an attempt to secure a Guinness World Record for the largest number of computers used to play a single game. Our hope is that public exposure will help to attract talented researchers, and that game researchers will consider adding distributed computing to their future research plans.

5. ACKNOWLEDGEMENTS

The authors would like to thank the ChessBrain team for their assistance in preparing this article. The ChessBrain project welcomes your involvement, and we maintain an open and public forum for discussion at:
http://www.ChessBrain.net.

To contact us visit: http://www.ChessBrain.net/friends.html

6. REFERENCES

AES-Rijndael (200X). National Institute of Standards and Technology:
http://csrc.nist.gov/CryptoToolkit/aes/

Beowulf (1994). Beowulf Introduction - History – Overview.
http://Beowulf.gsfc.nasa.gov/overview.html

Brockington, M. (1997). Asynchronous Parallel Game-Tree Search, Ph.D. Thesis. University of Alberta, Department of Computing Science.

Google Web Directory (2003).
http://directory.google.com/Top/Computers/Computer_Science/Distributed_ Computing/ Projects/

Lui, J. (2003). Distributed computing to tackle SARS patterns.
http://asia.cnet.com/newstech/
industry/0,39001143,39129226,00.htm

Oram, A. (ed.) (1998). Peer-to-Peer: Harnessing the Benefits of a Disruptive
Technology. O'Reilly & Associates, Inc. pp.75.

The New York Times (1997). Cracked Code Reveals Security Limits. http://distributed.net/pressroom/ mirror/nytimes-10-24-97.html

Zlib (200X). Zlib home page: http://www.gzip.org/zlib.