ChessBrain II – A Hierarchical Infrastructure for Distributed Inhomogeneous Speed-Critical Computation

Colin M. Frayn, Carlos Justiniano, Kevin Lew

***Abstract*—The ChessBrain project currently holds an official Guinness World Record for the largest number of computers used to play one single game of chess. In this paper, we cover the latest developments in the ChessBrain project, which now includes the use of a highly scalable, hierarchically distributed communications model.**

I. Introduction & Background

THE ChessBrain project was initially created to investigate the feasibility of massively distributed, inhomogeneous, speed-critical computation via the Internet. The game of chess lends itself extremely well to such an experiment by virtue of the innately parallel nature of game tree analysis, allowing many autonomous contributors to concurrently and independently evaluate segments of the game tree. With diminishing returns coming from increased search speed, we believe that distributed computation is a valuable avenue to pursue for all manner of substantial tree-search problems.

ChessBrain is among the class of applications which leverage volunteered distributed computing resources to address the need for considerable computing power. Earlier projects include the distributed.net (Prime number search) and the SETI@home project which is focused on the distributed analysis of radio signals.

Unlike similar projects which are content to receive processed results within days and weeks, ChessBrain requires feedback in real-time due to the presence of an actual time bound game. We believe that ChessBrain is the first project of its kind to address many of the challenges posed by stringent time limits in distributed calculations – a nearly ubiquitous feature of game-playing situations.

In the two years since ChessBrain played its first match, we have been working on a second generation framework into which we can host the same chess-playing AI structure, but which will enable us to make far better use of that same AI and will permit efficient access to a far wider range of contributors, including locally networked machines and dedicated compute clusters.

During the first demonstration match, the ChessBrain central server received work units from 2,070 machines in 56 different countries. Far more machines attempted to connect, but were unable to do so due to our reliance on a single central server. Our primary goal for ChessBrain II was to address this critical issue in a way that allowed for far greater scalability and removed much of the communication related processing overhead that was present in earlier versions.

As a result, we chose a hierarchical model, which we explain in detail in the following section. This model recursively distributes the workload thus freeing the central server from much of its prior time-consuming maintenance and communications management tasks.

II. Parallel Game Tree Search

We included the basic algorithms for parallel game tree search in our earlier papers[1,2,3], and they have been covered in detail in the literature. The ChessBrain project's core distributed search uses the APHID algorithm[4]. It implements an incremental, iterative deepening search, firstly locally on the server and then, after a certain fixed time, within the distribution loop. During this latter phase, the top few ply of the search tree are analysed repeatedly with new leaf nodes being distributed for analysis as soon as they arise. Information received from the distributed network is then incorporated into the search tree, with branches immediately being extended or pruned as necessary.

Leaf nodes are distributed to PeerNodes as work units. These encode the current position to be analysed and the depth to which it should be searched. Work units are distributed to the connected PeerNodes on a request basis, though they are also ranked in order of estimated complexity using intelligent extrapolation from their recorded complexity at previous, shallower depths. In this way, the most complex work units can be distributed to the most powerful PeerNodes. Work units that are estimated to be far too complex to be searched within a reasonable time are further subdivided by one ply, and the resulting, shallower child nodes are distributed instead. This is illustrated in figure 1.

Manuscript received December 17, 2005.

C. M. Frayn is with the Centre of Excellence for Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, UK. (e-mail: C.M.Frayn@cs.bham.ac.uk.)

C. Justiniano is a senior member of the Artificial Intelligence group at Countrywide Financial Corporation (CFC) by day and an independent researcher and open source contributor by night (e-mail: cjus@chessbrain.net)

Kevin Lew is a software architect at CCH. In his spare time he builds Beowulf Clusters. (e-mail: rawr@inorbit.com)

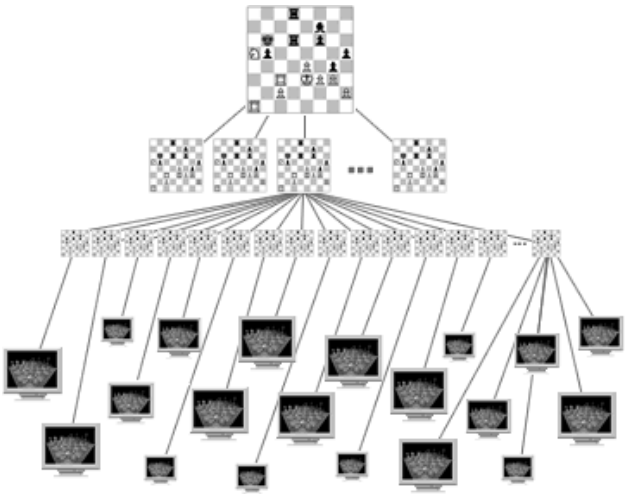

Fig. 1: Distributed chess tree search

If a node in the parent tree returns a fail-high (beta-cut) value from a PeerNode search, we then prune the remainder of the work units from that branch. This indicates that the position searched by the PeerNode proved very strong for the opponent, and therefore that the parent position should never have been allowed to arise. In this situation, we can cease analysis of the parent position and return an approximate upper limit for the score. PeerNodes working on these work units receive an abort signal, and they return immediately to retrieve a new, useful work unit.

III. CHESSBRAIN II

A. Motivation

The motivation behind ChessBrain II is to enable far greater scalability, whilst also improving the overall efficiency compared with the earlier version. Whilst ChessBrain I was able to support well over 2,000 remote machines, the lessons learned from the original design have enabled us to develop an improved infrastructure, which is suitable for a diverse range of applications.

B. Technical Configuration

ChessBrain II utilizes a custom server application, called msgCourier, which enables the construction of a hierarchical network topology that is designed to reduce network latency through the use of clustering as outlined in figure 2. The resulting topology introduces network hubs, the importance of which to graph theory has also been well covered in research superseding the random graph research of Erdos and Renyi and in the social network research of Milgram. In brief, well placed communications hubs help create small world effects which radically improve the effectiveness of networked communication. [5, 6].

The ChessBrain II system consists of three server applications, a SuperNode, ClusterNode and PeerNode.

Component	Purpose
SuperNode	Central server. Interfaces with the actual game being played. Manages work unit partitioning and distribution.
ClusterNode	Manages communities of local and distributed PeerNode servers.
PeerNode	Compute node servers. Performs work unit processing.

Table 1. Server Types

The central server no longer distributes work units directly to the PeerNodes, as was the case with ChessBrain I, instead work units are sent to an array of first-level ClusterNodes, operated by trusted community members. These ClusterNodes contain no chess-playing code and behave as network hubs (relay points) through which the complete set of work units can be passed.

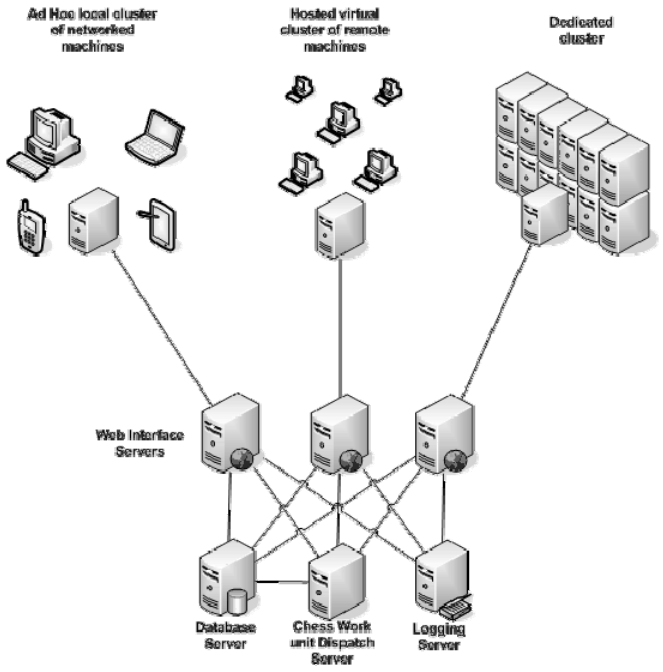

Fig. 2: ChessBrain II configuration

Each ClusterNode contains a complete listing of all PeerNodes connected to it, together with a profiling score to determine the approximate CPU speed of the PeerNode, exactly as in ChessBrain I. Each PeerNode connects to one and only one ClusterNode

The ClusterNodes, having been allocated a selection of individual work units by the SuperNode, then divide up these work units as they see fit based on the profiling data

that they obtain from their own network of PeerNodes. The primary considerations are that the work units are distributed to sufficient machines to ensure a reliable reply within the time required, plus to ensure that the work units perceived to require a greater computation effort are allocated to those PeerNodes deemed most fit to analyse them.

In subsequent versions, we intend to move some of the chess logic from the SuperNode onto the ClusterNodes, further reducing the communications overhead. Our anticipation is that the SuperNode will divide up the initial position into large tree chunks, and then distribute just these positions to the ClusterNodes. The ClusterNodes will then further subdivide the given positions, allocating the leaf nodes to the attached PeerNodes as it sees fit, and accumulating the returned results as and when they arrive. The ClusterNodes will then return a single result to the central SuperNode, instead of many.

C. ChessBrain II Communication Protocols

Early versions of ChessBrain relied on industry standard XML data encoding first using XMLRPC, and later using SOAP. The decision to use SOAP was driven by a desire for interoperability with emerging web services. However, the need to streamline communication has steered us toward minimizing our use of XML in favour of economical string based S-Expressions[7].

To further streamline communication we've implemented a compact communication protocol similar to the Session Initiation Protocol (SIP)[8] for use in LAN and cluster environments where we favour the use of connectionless UDP rather than stream-based TCP communication.

The ChessBrain I communication protocol consisted of XML content which was first compressed using ZLib compression and then encrypted using the AES Rijndael cipher. Although each PeerNode was quickly able to decrypt and decompress the payload content, the burden was clearly on the SuperNode server where each message to and from a PeerNode required encryption and compression operations. The situation was compounded by the fact that each PeerNode communication occurred directly with a single central SuperNode server.

With ChessBrain II we've eliminated direct PeerNode communication with the central SuperNode and introduced the concept of batch jobs, which combine multiple jobs into a single communication package. The reduction in messaging reduces the impact to the TCP stack while the grouping of jobs greatly improves the compression ratio.

D. Architecture Advantages

The most significant architectural change to ChessBrain involves the introduction of network hubs called ClusterNodes, as outlined in section IIIB.

ChessBrain I used a single SuperNode server to handle the remote coordination of hundreds of machines. Each dispatched job required a direct session involving the exchange of multiple messages between the SuperNode and its PeerNode clients. With ChessBrain II, jobs are distributed from a central server at distributedchess.net to remote ClusterNodes, which in turn manage local communities of PeerNodes. Each ClusterNode receives a batch of jobs, which it can directly dispatch to local PeerNodes thereby eliminating the need for individual PeerNode to communicate directly with the central server. This is necessary to harness a compute cluster effectively. Each ClusterNode collects completed jobs and batches them for return shipment to the central SuperNode server. The efficient use of ClusterNode hubs and job batching results in a reduced load on the central server, efficient use of clusters, reduced network lag, and improved fault tolerance.

We envision that ClusterNodes will largely be used by individuals desiring to cluster local machines. Indeed during the use of ChessBrain I we detected locally networked machines containing five to eighty machines. Most local networks in existence today support connection speeds between 10 to 1000 MBit per second, with the lower end of the spectrum devoted to wireless networks, and the higher end devoted to corporate networks, research networks and compute clusters. ChessBrain II is designed to utilise cluster machines by taking full advantage of local intranet network speeds and only using slower Internet connections to communicate with the SuperNode when necessary.

If we assume that there are roughly as many PeerNodes connected to each ClusterNode as there are ClusterNodes, then effectively the communications costs for each Cluster node, and indeed the SuperNode itself, is reduced to its square root. So, with total node count N, instead of one single bottleneck of size N, we now have approximately (sqrt(N)+1) bottlenecks, each of size sqrt(N). When addressing scalability issues, this is a definite advantage, allowing us to move from an effective node limit of approximately 2,000 to around one million machines.

E. Architecture Drawbacks

It is only fair to consider the drawbacks of the above architecture and to explain why it may not be suitable for every gaming application.

Firstly, as with any distributed computation environment, there is a substantial overhead introduced by remote communication. Indeed, communication costs increase as the number of available remote machines increases. ChessBrain I involved a single server solution that was overburdened as an unexpectedly large number of remote machines became available. Communication overhead on

ChessBrain I reached approximately one minute per move under peak conditions. However, with the experience gained since that first exhibition match, and with the subsequent redesign of ChessBrain I, we have reduced the overhead to less than ten seconds per move.

The presence of communication overhead means that shorter time scale games are not currently suitable for distributed computation. However, games that favour a higher quality of play over speed of play are likely to make good use of distributed computation.

Anyone who has ever attempted to write a fully-functioning alpha-beta pruning chess search algorithm featuring a multitude of unsafe pruning algorithms such as null-move, will immediately appreciate the complexity of debugging a search anomaly produced from a network of several thousand computers, each of which is running a number of local tree searches and returning their results asynchronously. Some of the complexities of such an approach are covered in [9].

Adding hierarchical distribution increases complexity, and highlights the importance of considering how a distributed application will be tested early in the design phase. With ChessBrain II we've had to build specialized testing applications in order to identify and correct serious flaws which might have otherwise proceeded undetected. Such a suite of testing tools is invaluable for a distributed application of this size.

F. Comparison with alternative parallel implementations

Other approaches towards parallelising search problems focus primarily on tightly-coupled compute clusters with shared memory. The aim of this paper is not to offer a thorough analysis of the advantages and drawbacks of remotely distributed search versus supercomputer or cluster-based search. The main advantages of this method over that used by, for example, the Deep Blue project [10] and the more recent Hydra project are as follows:

- Processing power – With many entirely separable applications, parallelising the search is a simple way to get extra processing power for very little extra overhead. For chess, the parallelisation procedure is highly inefficient when compared to serial search, but we chose this application because of its inherent difficulty, our own interest and its public image.
- Distributed memory – With many machines contributing to the search, the total memory of the system is increased massively. Though there is much repetition and redundancy, this still partly overcomes the extra practical barrier imposed by the finite size of a transposition table in conventional search.
- Availability – the framework described in this paper is applicable to a wide range of projects requiring substantial computing power. Not everyone has access to a supercomputer or a substantial Beowulf cluster.
- Costs – It's easier to appeal to 10,000 people to freely contribute resources than it is to convince one person to fund a 10,000 node cluster.

Drawbacks include:

- Communication overheads – time is lost in sending/receiving the results from PeerNodes.
- Loss of shared memory – In games such as chess, the use of shared memory for a transposition table is highly beneficial. Losing this (amongst other cases) introduces many overheads into the search time [11]
- Lack of control – the project manager has only a very limited control over whether or not the contributors choose to participate on any one occasion.
- Debugging – This becomes horrendously complicated, as explained above.
- Software support – The project managers must offer support on installing and configuring the software on remote machines.
- Vulnerability – The distributed network is vulnerable to attacks from hackers, and must also ensure that malicious PeerNode operators are unable to sabotage the search results.

At the present time, we are not aware of any other effort to evaluate game trees in a distributed style over the internet.

G. Comparison with other Chess projects

We are often asked to compare ChessBrain with more famous Chess machines such as Deep Blue and the more recent Hydra project. A direct comparison is particularly difficult as ChessBrain relies on considerably slower communication and commodity hardware. In contrast, both Deep Blue and Hydra are based on a hardware-assisted brute force approach. A more reasonable comparison would be between distributed chess applications running on GRIDs and distributed clusters.

H. The need for MsgCourier

While considering architectural requirements for ChessBrain II, we investigated a number of potential frameworks including the Berkeley Open Infrastructure for Network Computing (BOINC) project. BOINC is a software application platform designed to simplify the construction of public computing projects and is presently in use by the SETI@home project, CERN's Large Hadron Collider project and other high-profile distributed computing projects[12].

After extensive consideration we concluded that ChessBrain's unique requirements necessitated the

construction of a new underlying server application technology[13]. One of our requirements for ChessBrain II's software is that it must be a completely self-contained application that is free of external application dependencies. In addition, our solution must be available for use on both Microsoft Windows and Linux based servers, while requiring near zero configuration. The rationale behind these requirements is that ChessBrain II allows some of our contributors to host ClusterNode servers. It is critically important that our contributors feel comfortable with installing and operating the project software. We found that BOINC requires a greater level of system knowledge than we're realistically able to impose on our contributors. Lastly, BOINC was designed with a client and server methodology in mind, while our emerging requirements for ChessBrain II include Peer-to-Peer functionality.

Well over a year ago we began work on the Message Courier (msgCourier) application server in support of ChessBrain II. MsgCourier is designed to support speed critical computation using efficient network communication and enables clustering, which significantly improves overall efficiency. Unlike other technologies, msgCourier is designed to enable ad-hoc machine clusters and to leverage existing Beowulf clusters.

MsgCourier is a hybrid server application that combines message queuing, HTTP server and P2P features. When we embarked on this approach there were few such commercial server applications. Today, Microsoft has release SQL Server 2005 which combines a SQL Engine, HTTP server and messaging server features. The industry demands for performance necessitates the consideration of hybrid servers.

We chose to build msgCourier independently of ChessBrain (and free of chess related functionality) in the hopes that it would prove useful to other researchers.

The following were a few of our primary design considerations:

- A hybrid application server, combining message queuing and dispatching with support for store and forward functionality.
- Multithreaded concurrent connection server design able to support thousands of concurrent connections.
- High-speed message based communication using TCP and UDP transport.
- Built-in P2P functionality for self-organization and clustering, service adverting and subscribing.
- Ease of deployment with minimal configuration requirements.
- Built-in security features which are comparable to the use of SSL and or SSH.

The msgCourier project is under continued development. We are keen to emphasize here that the relevance of the ChessBrain project is not just to the specific field of computer chess, but to any distributed computation project. Hence, we believe that the msgCourier software is a valuable contribution to all areas of computationally intensive research. The direct application here demonstrates that the framework is also flexible enough to operate within gaming scenarios, where results are required on demand at high speed and with high fidelity, often in highly unpredictable search situations.

More information on the msgCourier project is available at http://www.msgcourier.com

CONCLUSIONS : THE FUTURE OF DISTRIBUTED GAMING

Distributed computation offers the potential for deeper game tree analysis for a variety of potential gaming applications. In particular, it overcomes the restrictions imposed by Moore's law, producing substantial gains for any game-playing code that is primarily computationally limited. For games such as Go, the effective contribution is reduced as the branching factor is so high that such games are algorithmically limited rather than computationally limited in most cases.

Speed-critical distributed computation also has many clear applications within the financial sector where rapid decisions must be made, often based on approximate or inadequate data.

During the past decade we've seen high profile Man vs. Machine exhibitions. We feel that the general public will eventually lose interest in exhibitions where a single human player competes against a machine which is virtually indistinguishable from the common personal desktop computer. Not since Deep Blue has any Man vs. Machine event really captured the public's imagination.

We feel strongly that the future of Man vs. Machine competitions will migrate toward a format where a human team competes against a distributed network. Such events will take place over the Internet with distributed human members collaborating remotely from their native countries. This exhibition format will likely capture the public's imagination as it more closely resembles themes played out in popular science fiction.

On the ChessBrain project we've learned the importance of capturing the public's imagination for without their support massively distributed computation would not be economically feasible[14]. Generally, a project is only as good as the contributors that it is able to attract. This entire field of research – that of attracting distributed computation teams to a project – seems remarkably underdeveloped in the literature, despite the fact that it has an arguably greater effect on the success of any distributed project than any

degree of algorithmic sophistication. More work in this area seems extremely important, though it lies firmly within the realms of psychology and sociology rather than pure computer science.

We've completed preliminary testing on small clusters with the support of ChessBrain community members [15]. During the first quarter of 2006 we intend to release a major update of our project software when we will begin large-scale public testing of ChessBrain II. We expect ChessBrain II to be fully operational by the second quarter of 2006.

We are actively preparing for a second demonstration match between ChessBrain II and a leading international chess grandmaster within the next 12 months. Anyone wishing to contribute to this event is welcome to contact the authors at the addresses supplied.

ACKNOWLEDGMENT

We would like to acknowledge the hundreds of international contributors who have supported the ChessBrain project over the past four years. In addition, and by name, we would like to thank Cedric Griss and the Distributed Computing Foundation; Kenneth Geisshirt and the Danish Unix users Group; Peter Wilson; Gavin M. Roy with EHPG Networks; and Y3K Secure Enterprise Software Inc., whose outstanding support enabled us to establish a world record and to further contribute to this emerging field. Colin Frayn is currently supported by a grant from Advantage West Midlands (UK).

REFERENCES

[1] Frayn, C.M. & Justiniano, C., "The ChessBrain Project – Massively Distributed Inhomogeneous Speed-Critical Computation", Proceedings IC-SEC, Singapore, 2004

[2] Justiniano, C. & Frayn, C.M. "The ChessBrain Project: A Global Effort To Build The World's Largest Chess SuperComputer", 2003 ICGA Journal, Vol. 26, No. 2, 132-138

[3] Justiniano, C. "ChessBrain: A Linux-Based Distributed Computing Experiment", 2003 Linux Journal, September 2003

[4] Brockington, M. "Asynchronous Parallel Game-Tree Search", 1997 PhD Thesis, University of Alberta, Dept. of Computer Science

[5] Barabasi, A-L. "Linked: The new Science of Networks", 2002. Cambridge, MA: Perseus

[6] Gladwell, M. "The Tipping Point", 2000. Boston: Little and Company

[7] Rivest, R. L., "S-Expressions"., MIT Theory group, http://theory.lcs.mit.edu/~rivest/sexp.txt

[8] Session Initiation Protocol (SIP) http://www.cs.columbia.edu/sip/

[9] Feldmann, R., Mysliwietz, P., Monien, B., "A Fully Distributed Chess Program", Advances in Computer Chess 6, 1991

[10] Campbell, M., Joseph Hoane Jr., A., Hsu, F., "Deep Blue", Artificial Intelligence 134 (1-2), 2002.

[11] Feldmann, R., Mysliwietz, P., Monien, B., "Studying overheads in massively parallel MIN/MAX-tree evaluation", Proc. 6th annual ACM symposium on Parallel algorithms and architectures , 1994

[12] Berkeley Open Infrastructure for Network Computing (BOINC) project: http://boinc.berkeley.edu/

[13] Justiniano, C., "Tapping the Matrix: Revisited", BoF LinuxForum, Copenhagen 2005

[14] Justiniano, C., "Tapping the Matrix", 2004, O'Reilly Network Technical Articles, 16th, 23rd April 2004. http://www.oreillynet.com/pub/au/1812

[15] Lew, K., Justiniano, C., Frayn, C.M., "Early experiences with clusters and compute farms in ChessBrain II". BoF LinuxForum, Copenhagen 2005