

Hazel Mckendrick

Project Proposal

Distributed Computing for Virtual Worlds

Supervised By Dr. Henry Fortuna

## **Introduction**

---

Fundamental to computer hardware and software developments, Moore's law states that the number of transistors that can be fitted onto an integrated circuit increases exponentially over time. From the 1970s to the present day, this has for the most part held true, with the number of transistors doubling approximately every two years. While previously this could be related to gains in CPU clock speeds, leading directly to speed increases for sequential programs, in a 2004 paper Sutter demonstrated that this is no longer possible; 'The free lunch is over'.

Physical limitations mean that CPU development now tends towards a multi-core approach, which only parallel software can truly realise the benefits of. With almost 80% of video game player's computers sporting dual-core or quad-core CPUs (Valve Corporation 2009) and consoles tailored towards parallel computation, it becomes clear that concurrency is integral to future performance gains in games development and software development in general.

Few subject areas in games demonstrate this more clearly than the development of servers for MMOs (Massively Multi-player Online games) and persistent worlds. As well as connecting to client applications across the planet, servers themselves are commonly distributed over multiple nodes to update characters, connections, entities and the world itself. As Waldo (2008) explains:

'...any online game or virtual world will involve a large number of servers (or will have failed so miserably that no one either can or wants to remember the game or world). '

## Motivation

---

In order to handle large scale problems in a timely fashion, exploiting equally large-scale hardware seems the obvious solution. In the past this has involved mainframe servers and even high performance super-computers, however a single system approach has limitations. Expansion can be challenging, costly or impossible, and of course a single computer system cannot readily be split over multiple geographic locations.

These are areas where distributed computing systems bear a distinct advantage; systems can be positioned within a single location or distributed across the world, even across entirely different types of hardware as demonstrated by the Folding@Home project. Additional nodes can be added when loads on the system are high, and shut-down when they are no longer required. Consequently improving performance is important not only to create an efficient system, but also to reduce running costs.

Relating this to games, Esbensen (2005) states that:

'The current infrastructure development and maintenance costs of MMORPGs are astronomical.' but offers methods to improve server node throughput and reduce lag, claiming:

'...the implementation of a beautifully efficient, technologically sound online gaming infrastructure-which optimizes speed and cuts costs by reducing unnecessary data transmissions-is a viable solution. '

Distributed computing for games faces additional problems when compared to many commercial and scientific applications in that, like all real-time systems, strict temporal constraints must be adhered to. While a system such as Pixar's rendering farms can allow hours to process a single frame (Christensen 2006), "soft" real time systems such as games require a consistently fast response and must be able to recover from and accommodate for situations where this is not the case; data transfer, synchronisation and response times are all fundamentally important.

Although network latency may not be as critical to MMOs as to fast-paced action games (Beigbeder 2004), the ability to remain playable at high latencies relies on strategies within an

individual game (Fritsch 2005) and as Waldo (2008) summarises:

'...the most important goal for a game or virtual world is that it be fun. ... Latency is the enemy of fun—and therefore the enemy of online games and virtual worlds.'

At their most basic, servers for virtual worlds deal with a geographic expanse of virtual land, and the character entities within it. By considering these tasks alone and dealing with a distributed system of servers rather than player clients which might connect to them, it will be possible to seek performance gains through intelligent spatial partitioning and workload balancing and to investigate scalability and varying approaches to parallel decomposition of the problem. This simulation relates to a range of situations out with games, and offers various opportunities for visualisation, expansion and extension.

## Research Question

---

*'How can the processing of autonomous characters in a real-time virtual environment benefit from distribution over multiple computer systems?'*

## Literature Review

---

Currently, approaches to parallelism in games vary widely between developers and titles. While El Rhalibi, Costa and England (2005) propose a 'Concurrent Game Programming Framework' to provide tools for the production of parallel applications, games must be modelled as cyclic-task-dependency graphs in order to make use of it—no small task in a field where designs and requirements are constantly changing.

At the opposite end of the spectrum exist frameworks for creating online games, of which there are several examples with differing approaches but similar ideals, aiming to abstract explicit parallelism away from the developers using them. Examples of these include Project Darkstar (Sun Microsystems 2007), which aims to process tasks without a need for multiple “shards”—mirrors of the game world occupied by separate sets of players—or “zones”—areas of the game world handled by a separate server node. Hydra (Chan et al.) demonstrates a peer-to-peer approach to the same problem, but exhibits the scalability issues associated with this architecture, an area where client-

server modelled Project Darkstar boasts particular strength. While hiding the issues of distributed computing or peer-to-peer networking seem ideal, Wollrath et al (2004) argue that programmers must be aware of the intrinsic differences between these and shared memory systems during development in order to construct correct and efficient applications.

Whether or not a framework is used, a node based client-server approach appears is a typical model for MMO servers. Assiotis and Tzanov (2006) demonstrate such a method, using a system of events, areas of interest and regional or object locks to safely distribute and process work. Although yet to be seen from a successful commercial MMO, much research has also been conducted into peer-to-peer models. Iimura, Hazezama and Kadobayashi (2004) suggested a world divided into zones with individual owners could mitigate hardware costs for developers, but would also complicate access, security and redundancy.

Where a problem is processed over multiple machines, consideration must be given as to how it is divided and how workload is balanced between system. Depending on whether a simulation is fully 3d or based on a 2d grid, BSP trees, quadtrees, octrees and kd-trees can all be relevant structures for spatially partitioning entities in a world. De Vleeschauwer et al. (2005) proposed a technique, employed by Assiotis and Tzanov (2006), where areas are divided into microcells. Each computer system process a fixed or dynamic set of these, chosen by a deployment algorithm, with entities free to move between them.

## **Methodology**

---

The overall aim of the project is to answer the aforementioned research question by producing applications to allow characters to be processed across several computer system “nodes”, and by evaluating whether the approaches used lead to gains in performance over alternative techniques.

The simulation will involve a number of characters, represented by simple state machines, navigating and performing simple tasks in a 2D grid-based world. This world will be split across available nodes, statically at first but dynamically if time allows, and characters will be transferred between these as they move around. As De Vleeschauwer et al.(2005) recommend, events occurring at a boundary between two divided areas will be forwarded from one node to another.

Initially, characters will have no knowledge of each other, however a possible extension to the project would be to allow for collisions or interactions. This will be considered if sufficient time is available, however the data interdependencies would greatly complicate parallelism in the simulation.

Following on from Assiotis and Tzanov (2006) the project will use a client-server architecture, with one “master” node controlling the work of others. A hybrid approach will be taken to parallelism as tasks will be distributed between independent computer systems as well as between different processor cores within one computer. As the problem at hand lends itself to finding parallelism through task decomposition, a thread-pooling technique will be used locally, which also necessitates investigation into task scheduling. A message passing approach will be used between nodes.

In terms of specific technologies, several have been considered: OpenMP allows for low-level, user-directed parallelism, suited to the adaptation of serial programs in a shared memory system; MPI (Message Passing Interface) is the *de facto* standard for High Performance Computing, allowing communication between systems but also dictating a message passing approach to application design; CORBA (Common Object Request Broker Architecture) allows access to objects across multiple systems and platforms and was found by Broekhuizen (2004) to be suitable for games despite its lack of use in this field. The .NET framework is to be used for this project due to its flexibility, clarity of networking and multi-threading support and increased use within the games industry. As this already provides a range of synchronisation primitives it caters for parallelism within a single computer system, and MPI with TCP-IP will be used for communication between processes and nodes.

## **Evaluation**

---

The intended deliverables for the project are client and server applications which allow the server to distribute work between multiple client nodes. The project will be considered successful, regardless of performance, so long as these are produced and conclusions or recommendations can be formulated concerning the approaches employed.

Practical and theoretical performance in terms of throughput, speed-up and serial fraction will be considered subject to several criteria. These include varying the number of client nodes in the simulation, differing approaches to spatial partitioning and work distribution, parallelism within the client application and message passing techniques. While the first criterion is fundamental to the project, the others may require additional implementation work and will be investigated subject to time and perceived importance.

In order to reach more meaningful conclusions, measurements taken will be considered with reference to several established theories: Amdahl's Law, Gustafson's Law and the Karp-Flatt Metric. Amdahl, a proponent of single processor systems, highlights several problems with parallelism (Amdahl 1967) and gives the theoretical maximum speed gain of using multiple processors given the percentage of the problem which can be processed in parallel  $P$ , the percentage which must be processed in sequence  $S$ , and the number of processors  $N$  as

$$\text{Max. Speed Gain} = \frac{1}{S + \frac{P}{N}} \quad (\text{where } S = 1 - P)$$

As  $N$  tends to infinity, the maximum speed gain tends to  $1 / S$ , demonstrating that the overall performance gain is limited by the part of the problem which cannot be processed in parallel. Hill and Marty (2008) showed that despite recent hardware changes Amdahl's Law is still valid, and that minimising and measuring  $S$  is of critical importance.

Gustafson's Law assumes that any parallel problem to be solved will be scaled in-line with the number of processors available (Gustafson 1988), stating that any appropriately large problem can be parallelised efficiently. In effect, as  $N$  tends to infinity  $P$  is also scaled towards infinity, giving

$$\text{Max. Speed Gain} = N - (N - 1) \times S$$

As the simulation proposed does not have limitations on the data set to be processed, it will be relevant to investigate this law as the number of client nodes is increased.

Finally, the Karp-Flatt determines a serial fraction  $e$  which gives a numerical representation of how well a system has been parallelised (Karp and Flatt 1990). Consistent with Amdahl's Law, this metric uses speed up  $\psi$  and number of processors  $p$  to give

$$e = \frac{\frac{1}{\psi} - \frac{1}{p}}{1 - \frac{1}{p}}$$

In this way, overhead from parallel processing (such as managing threads) which might not be considered with the sequential part of a program when using Amdal's or Gustafson's laws can be taken into account. As well as its use in the final evaluation of the project, this metric may also be employed continuously during development as a diagnostic tool.

## Schedule

---

The intended schedule for completion of the project is detailed in the following Gantt chart. These time-frames accommodate for the fact that final deadlines have not yet been specified by allocating flexible periods of time for expansion and evaluation.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
<b>Task</b>	18/01/10	25/01/10	01/02/10	08/02/10	15/02/10	22/02/10	01/03/10
Final Structure Planning							
Character Data Structures							
Character Navigation and Tasks							
Task Scheduling and Thread Pool							
Client-Server Networking							
Server Task Distribution							
Client Task Processing							

	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16
<b>Task</b>	08/03/10	15/03/10	22/03/10	29/03/10	05/04/10	12/04/10	19/04/10	26/04/10	03/05/10
Extension and Expansion									
Testing and Evaluation									
Write Dissertation									
Presentation									

## Requirements

---

As the project investigates distributed computing, multiple computer systems will be required for testing at points throughout the project, although it should be possible to develop on just one. While a Beowulf cluster of computers running Linux or a Unix-like operating system would be ideal for this project, computers running Microsoft Windows XP in the university's Game

Lab are also well suited and easily accessible; as such these will be used. Corresponding with the project's requirements, these machines feature multi-core CPUs and are on a shared local area network.

In terms of software, Microsoft's .NET framework and Visual Studio 2008 will be required; these are already present on the computers I intend to use. Additionally, the MPI.NET library will need to be installed which requires either either Microsoft's Compute Cluster Pack or HPC Pack to provide use of and headers for the MS-MPI implementation.

## References

---

- Amdahl, G. M. 1967. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* (Atlantic City, New Jersey, April 18 - 20, 1967). AFIPS '67 (Spring). ACM, New York, NY, pp. 483-485
- Assiotis, M. and Tzanov, V. 2006. A distributed architecture for MMORPG. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support For Games* (Singapore, October 30 - 31, 2006). NetGames '06. ACM, New York, NY. p. 4
- Beigbeder, T. et al. 2004. The effects of loss and latency on user performance in Unreal Tournament 2003. In *Proceedings of ACM Network and System Support for Games (NetGames) Workshop* (Portland, OR, Sept. Sept. 2004). ACM Press, New York, 2004, pp. 144–151.
- Chan, L. et al. 2007. Hydra: a massively-multiplayer peer-to-peer architecture for the game developer. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support For Games* (Melbourne, Australia, September 19 - 20, 2007). NetGames '07. ACM, New York, NY, pp. 37-42
- Christensen P. H, et al. 2006. Ray tracing for the movie 'Cars'. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006* (Salt Lake City USA September 18-20 2006) IEEE. pp.1-6.
- Esbensen, B. 2005. [online] *Online game architectures: Back-end strategies*. Available from: [http://www.gamasutra.com/gdc2005/features/20050310/esbensen\\_01.shtml](http://www.gamasutra.com/gdc2005/features/20050310/esbensen_01.shtml) [Accessed October 2009]
- Fritsch, T., Ritter, H., and Schiller, J. The effect of latency and network limitations on MMORPGs: A field study of Everquest 2. In *Proceedings of the Fourth ACM Network and System Support for Games (NetGames) Workshop* (Hawthorne, NY, Oct. 10–11). ACM Press, New York, 2005.
- Gustafson, J. L. 1988. Reevaluating Amdahl's law. *Commun. ACM* 31(5), 532-533.
- Hill, M. and Marty, M. 2008. Amdahl's law in the multicore era. *Computer*, 41(7), p. 33
- Iimura, T., Hazeyama, H., and Kadobayashi, Y. 2004. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support For Games* (Portland, Oregon, USA, August 30 - 30, 2004). NetGames '04. ACM, New York, NY, pp. 116-120
- Karp, A. H. and Flatt, H. P. 1990. Measuring parallel processor performance. *Commun. ACM*, 33(5), pp. 539-543.
- El Rhalibi, A., Costa, S. and England, D. 2005. Game Engineering for a Multiprocessor Architecture. *Digital Games Research Conference*, (Vancouver, British Columbia, Canada June 16-20, 2005). Changing Views: Worlds in Play.

Sun Microsystems 2007. [online] *Project Darkstar: Changing the game*. Available from: [http://projectdarkstar.com/w/images/3/35/Project\\_Darkstar\\_Changing\\_the\\_Game.pdf](http://projectdarkstar.com/w/images/3/35/Project_Darkstar_Changing_the_Game.pdf) [Accessed November 2009]

Sutter, H. 2004. *The free lunch is over*. [online] Available from: <http://www.gotw.ca/publications/concurrency-ddj.htm> [Accessed November 2009]

Valve Corporation. 2009. *Steam Hardware Survey* [online] Available from: <http://store.steampowered.com/hwsurvey/> [Accessed December 2009]

De Vleeschauwer, B. et al. 2005. Dynamic microcell assignment for massively multiplayer online gaming. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support For Games* (Hawthorne, NY, October 10 - 11, 2005). NetGames '05. ACM, New York, NY, pp. 1-7.

Waldo, J. 2008. Scaling in games & virtual worlds. *ACM Queue*. 6(7), pp.10-16.

Wollrath et al. 2004. [online] *Note on Distributed Computing*. Available from: <http://research.sun.com/techrep/1994/abstract-29.html> [Accessed December 2009]

## **Bibliography**

---

Andrews, G. 2000. *Foundations of multithreaded, parallel and distributed programming*. Addison Wesley.

Caltagirone, S. et al. 2002. Architecture for a massively multiplayer online role playing game engine. *J. Comput. Small Coll.* 18(2), pp. 105-116.

Cooling, J. E. 1991. *Software design for real-time systems*. Chapman & Hall. pp.285-328

Mattson, T., Sanders, B. and Massingill, B. 2004. *Patterns for parallel programming*. Addison Wesley.

Moore, G. 1965. *Cramming more components onto integrated circuits*. *Electronics*, 38(8).