PIKKOTEKK

TECHNICAL SUMMARY

FEBRUARY 2011

» INTRODUCTION

» WHAT DOES PIKKO SERVER DO?

The defining feature of the Pikko Server software is the ability to allow an online game to handle a nearly limitless number of players in the same location in a virtual world. The number of players you can see in one battlefield or in one city square is only limited by the client's capabilities. Our server side software product makes it possible to develop high player density games on existing game engines, increasing their scale and offering vast new possibilities.

» HOW DOES IT WORK?

The Pikko architecture consists of Pikko Server and several cell servers working together. Players in an online multiplayer game connect to Pikko Server, which handles load balancing between the cell servers. The cell servers handle physics, game logic and more.

Each cell server handles only the activity in a small part of the virtual world. The cell servers can be seen as cells in a virtual mobile phone network. When a person with a mobile phone moves in the real world, the phone will switch to the closest base station without the mobile phone user noticing it. In the same way, a player moving around in a virtual game world will switch completely seamlessly between the different cell servers.

By using tried and tested software technology from the telecom industry, Pikko Server enables the development of truly scalable, high-player density games.

» ENGINE INTEGRATION

» WHAT PIKKO SERVER DOES

The client and the cell server shown in figure 1 can be developed using any game engine such as Unity or Unreal Engine. The cell server is basically a normal game server instance. For example, game developers that have already built a multiplayer game using Unity can very easily transform that game into a massively multiplayer online (MMO) game by using Pikko Server. The time it takes to integrate Pikko Server into your existing game depends on the game's characteristics, but it is a matter of weeks, not months or years. PikkoTekk experts will help during this transition. Unity is the first engine that the PikkoTekk technology is integrated into. The integration was released in February 2011.

» THE CLIENT

The client needs minimal or no modification when you start using Pikko Server. The client is not aware of the number of cell servers. It just connects to the cluster via one IP address and after that, all the traffic from it will pass through Pikko Server. Both the UDP and TCP transport protocols are supported.

» THE CELL SERVER

The cell server is a classic game server built by a game developer using a game engine. These cell servers are connected to Pikko Server but are not themselves products from PikkoTekk. The cell servers will need some minor modifications to be a part of

a Pikko Server cluster. Pikko Server needs to be able to exchange player object information between cell servers. To transform a normal game server into a cell server, it needs an API for sending and receiving object states. This is usually a minor task. All game server instances are already capable of sending object states to the game clients, and the same mechanism can be reused by Pikko Server for sending object states between cell servers. When Pikko Server decides that a player enters the cell server's area of responsibility, the server will accept serialized data for the new player, in a fast and reliable way. You can read more about the handover technology further down in this document.

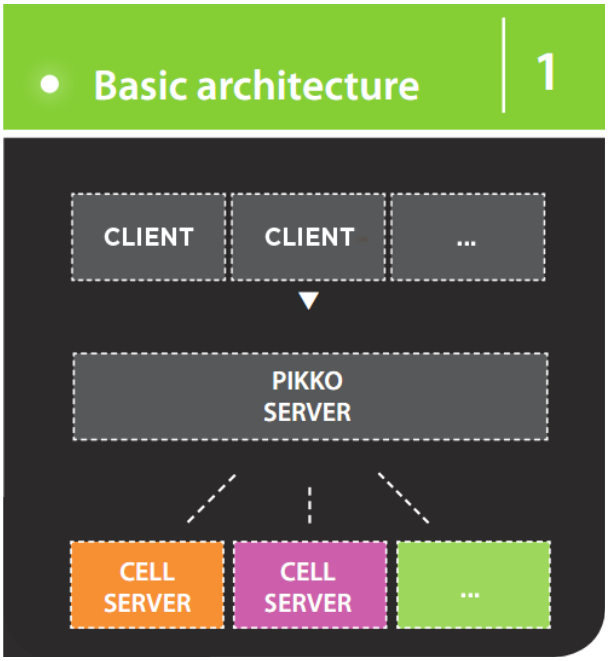

Figure 1: The basic architecture of a Pikko Server setup, showing the clients connecting to Pikko Server, which in turn routes communcation to the cell servers.

» TECHNICAL DESIGN

» A DISTRIBUTED VIRTUAL WORLD

Pikko Server works by distributing a virtual world over several cell servers, giving each a dynamic part of the world to handle. At the same time, it ensures that players will be given a single, coherent world view that may span over many cell servers. This is accomplished by Pikko Server acting as a load-balancer for the network traffic in the game. After clients and cell servers have connected to Pikko Server, all traffic between them will go through Pikko Server, which will intelligently decide how to route the traffic in order to divide the load between cell servers in the best possible way.

Pikko Server is aware of all the connected players and their positions in the virtual world, and of how the world is divided among the cell servers. Depending on the player's current position, the traffic from it will be routed to the cell server it is currently in. That player will be getting information from its current cell server, as well as from other cell servers in the player's vicinity. By compiling together outgoing data from several cell servers, players are able to see the actions taking place in other cell servers, making it look like a single, massive world.

» HANDOVER OF OBJECT STATE

When a player, or any other object in the virtual world, moves away from a cell server's area into another cell server, Pikko Server hands that object over from the old server to the new server, as shown in figure 3. This handover is both fast and seamless, and - since it is only made on the server

side - completely transparent to the client. Handovers are dictated by Pikko Server and initiated automatically when objects cross over cell server boundaries.

The easiest way to understand handovers is to consider the case when a player in one cell server moves towards the boundary of an adjacent cell server. At that point, Pikko Server requests the player's state from the player's current server and receives it in serialized form. This state can include, for example, the player's position, posture, equipment, movement direction, speed and rotation. The serialized state is then sent to the new cell server, which can now start handling the game logic of the player. Finally, Pikko Server tells the player's original server to remove its instance of the object.

» THE MAST ALGORITHM

Pikko Server uses a unique load balancing technology inspired by the cell phone masts used in cellular networks. Each cell server has an associated «mast» in the virtual world that determines which players it is responsible for. A player is then handled by the cell server with the closest mast. As players move around in the virtual world, they are automatically and seamlessly switched and handed over between cell servers in the same way a cell phone is switched to the nearest physical mast. However, Pikko Server goes one step further by letting the virtual masts themselves move around in the virtual world according to the current game situation, in order to always yield an optimal division of the game objects to the different cell servers. If a cell server becomes overloaded with players, its mast will reposition itself and handovers will be made to nearby servers automatically to accommodate the situation and balance the load.

Because of this dynamic nature of the masts, cell servers can even be added or removed automatically by Pikko Server in runtime to fit the current load. This results in an unprecedented level of scalability that is not dependent on player actions and movement, relieving the game developer of the need to divide the virtual world into static zones or limit their gameplay, and enabling them to build truly limitless open-world environments.

» PROXY OBJECTS

To ensure that interactions taking place near the borders of different cell servers work correctly, Pikko Server makes use of so-called proxy objects. When an object moves close to its cell server's border, a proxy object representing that object will be spawned on the neighbouring cell server. This proxy object acts as a mirror of the original object, getting the same state updates and interacting with local objects in its vicinity. This means that objects on the proxy's server can interact with the proxy object, for example colliding with it, and the proxy can then communicate with the original object to let it know of important events. The proxy mechanism is essential for creating seamless game server boundaries while maintaining well-defined, consistent interactions on the borders.

» INTERCELL COMMUNICATION

For interactions taking place over larger regions, spanning several cell servers, proxy objects are not sufficient. A common example is a player shooting with a gun, using a raycast to determine which opponent gets hit. As a cell server can only see objects in a limited region, it may not be able to directly access the game object of the opponent that was hit. For these situations, Pikko Server features special types of communication between cell servers to be able to forward player actions across cell server boundaries.

Using this feature, the example can be solved by having the original server forward the raycast to the next server along the raycast's line, letting this server continue the calculation in its region. Raycasting is only one of several types of intercell communication supported by Pikko Server. Another example is area effects crossing one or several borders.

» HORIZONTAL SCALABILITY

By distributing the world over many servers, a game using Pikko Server can efficiently utilize today's multicore processor architectures. Since the server-side is parallelized on a per-cell-server basis, each cell server can be put on one core each. Thus, instead of using complex multi-threading techniques to get maximum usage of the cores, cell servers can be programmed single-threaded and identical to the other cell servers, greatly simplifying the task of programming them. You get all the benefits of a distributed system while keeping the ease of programming one single server.

The result of this is that a game using Pikko Server scales horizontally. That is, instead of increasing the hardware power of existing servers to cope with increasing load, more servers and more machines can be added. This leads to vastly improved scalability, redundancy, and maintainability.

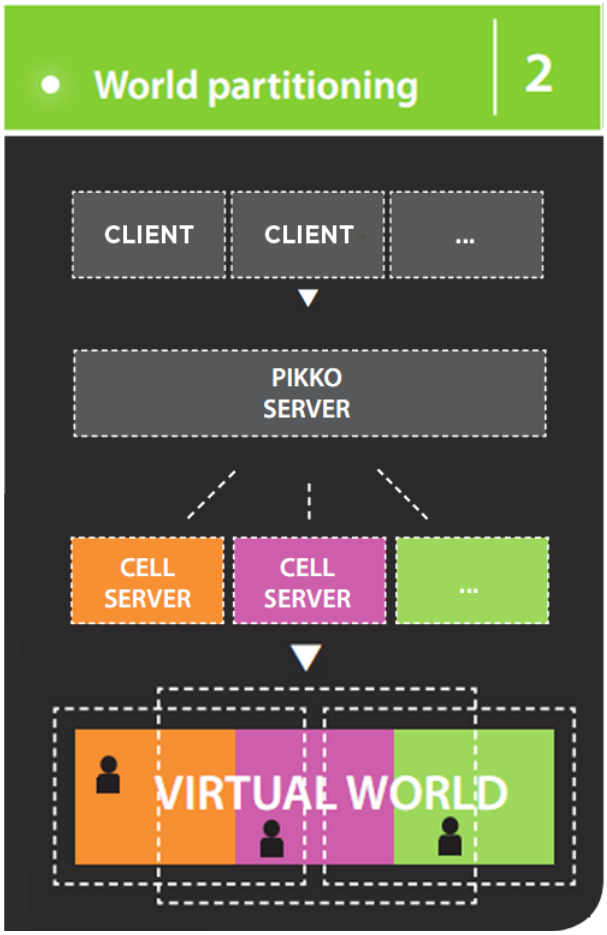

Figure 2: An example showing three cell servers with players in them. Each player is handled by the cell server that has its origo closest to it. The dashed lines around each cell server represent the area within which proxy objects are spawned on that server.

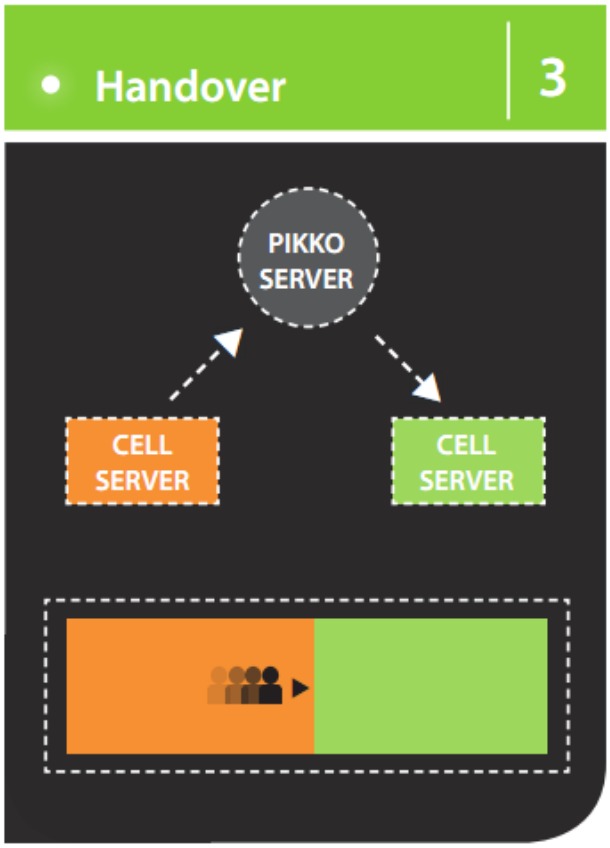

Figure 3: Pikko prepares the handover when a player is close to a cell server border. When a player moves from one cell server to another, Pikko requests the player state from the exiting cell server and sends it forward to the arriving cell server. The player state is then removed from the exited cell server.

» TELECOM TECHNOLOGY

Pikko Server is written in Erlang, a concurrent programming language and runtime system developed by the telecom industry (Ericsson). This allows Pikko Server to be scalable and robust under massive workloads. Thanks to the powerful parallel architecture of Erlang, Pikko Server can handle large amounts of network traffic both efficiently and safely. Because processor manufacurers are building more cores into each processor instead of increasing the clock frequency, a parallel software architecture is needed in order to draw the benefits of this development. Erlang is a platform built from the ground up for parallel computing and can use modern multicore processors very efficiently. Together with Pikko Server's architecture, it turns any single-core game server into a cluster of cell servers, drawing maximum utilization out of the hardware's capacity.

» FAILOVER

Pikko Server's powerful redundancy features ensure that games stay alive even in the event of a fatal crash in a cell server. Thanks to the dynamic features of the mast algorithm and the proxy mechanism, Pikko Server is perfectly equipped for handling failover scenarios gracefully. By guaranteeing that each game object is always present on several cell servers, the amount of damage incurred from a crash is kept to a minimum and only localised to the affected area in the virtual world. When a cell server goes down, the surrounding cell servers will automatically take over responsibility for the objects in that region. Later, a new cell server can be seamlessly launched and begin handling objects to lower the load on the other cell servers.

These features are essential for today's demanding, persistent MMO games with players playing around the clock, where reliability is crucial. Since client connections are never lost during a failover, players can continue playing like nothing ever happened.

» CONTENT PUBLISHING

The redundancy features of Pikko Server are not only geared at solving failover situations - they can also be used for updating server code and content in runtime. At any time, a cell server can be gracefully stopped by an operator without disrupting the game. The objects handled by that server will then be handed over to adjacent cell servers, giving them the main responsibility for the objects. The stopped cell server can now be updated with new code, perhaps for fixing a bug, or with new content to be made available to the players. When it is ready, the cell server can be restarted again.

By repeating this process for all the cell servers in the cluster, changes can be deployed across the whole system without ever stopping the game and without any player being thrown out. This makes it incredibly easy to release bug fixes and content updates while maintaining high availability, giving players more time to enjoy their game.

» GAME INTEGRATION

There are several ways to integrate Pikko Server into a new or existing game. Depending on the game's needs, Pikko Server can either be used for governing the entire game world or just a part of it. For example, in an MMO with many different zones, where each zone is located on a separate server, Pikko Server can be used for load-balancing a particularly popular zone. This zone can then seamlessly be distributed over several subservers while still acting as a single zone to the outside world. In this case, the rest of the servers stay untouched and need not even be aware that Pikko Server is used for one of the zones. This flexibility means that the initial threshold for integrating Pikko Server into an existing game is very low - it can be used only where it's needed.

In addition to load-balancing individual zones, Pikko Server can be used to eliminate zoning, partly or completely. Static zoning is itself an attempt to load-balance a game by dividing it into zones run on different servers - the very problem that Pikko Server solves using dynamic and transparent zones. This means that a game using static zoning can merge together a set of zones into single, larger zones and remove potential boundaries between them. Each such zone can then be handled by Pikko Server, which will internally divide it between several servers. This approach provides many benefits. One is that players will see the world as a single contiguous area and will not have to endure load times when switching between zones. Another benefit is that the number of cell servers handling a large zone can be changed in runtime to fit the current load, giving drastically increased scalability.

» MONITORING TOOLS

When a massive multiplayer game is released and running, the monitoring of the game and its internal state is crucial. Pikko Server comes with a set of graphical web based tools that can show recorded and stored data from the game. An example of a useful report is a 2D-map with playback, showing the cell server partitioning and the coordinates of the players. For more information about other reports such as heatmaps and network-usage graphs, please

contact our technical staff. The graphical tools are upgraded and developed separately from the main Pikko Server. They will also be released as open source. This makes it easier to add customized graphical representation for any of the available measurements in the cell servers or the Pikko Server itself. Pikko Server is also capable of recording network traffic to be able to analyze crashes and exceptions in a test and debug environment.

» FURTHER INFORMATION

Please contact us for more details. We are happy to answer any and all questions you might have.

» CEO - Christian Lönnholm
christian.lonnholm@pikkotekk.com
+46 - 733 99 38 44

» CTO - David Almroth
david.almroth@pikkotekk.com
+46 - 70 417 46 10