Guild Wars 2:
Scaling from one to millions

Stephen Clarke-Willson, Ph.D.
Server Programming Team Lead &
Interim Studio Technical Director
ArenaNet

GAME DEVELOPERS CONFERENCE® 2013
SAN FRANCISCO, CA
MARCH 25–29, 2013
EXPO DATES: MARCH 27–29

Puppies!

This is not a stock photo. These are the siblings of our puppy. Aren't they cute? I'm told presenting a picture of something cute at the beginning of a talk will generate warm feelings in the audience, and who doesn't want that?

Outline of the talk

Two parts:

Some background on ArenaNet and the company goals.

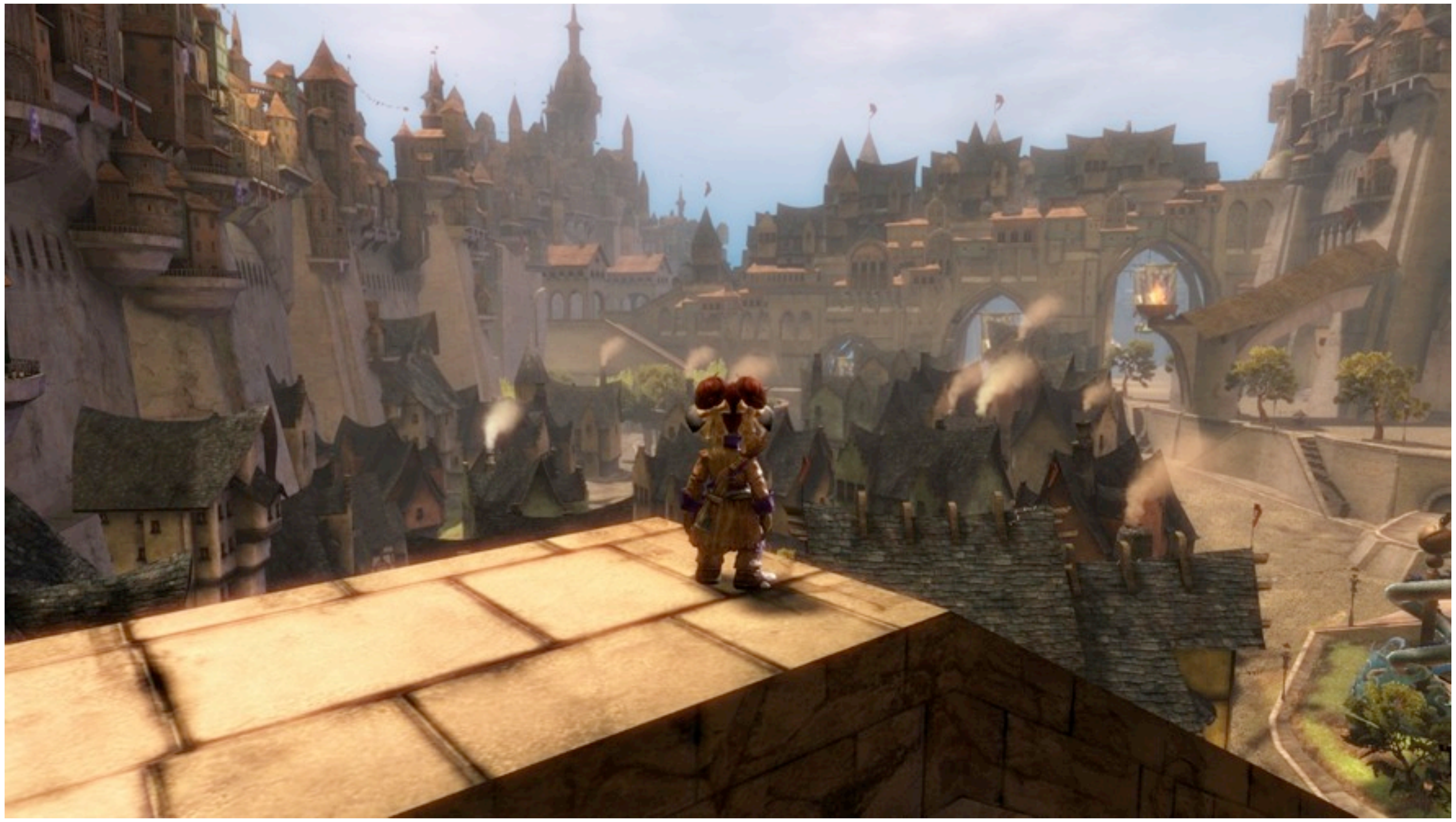
Some user stories (in the Agile sense) .

But first ...

Screenshots!

I skimmed through these in a few seconds during the talk; they are really just here to provide some context in case someone looks at the slides online and has no idea what Guild Wars 2 looks like. Visit http://guildwars2.com for the latest scoop.

Screenshots



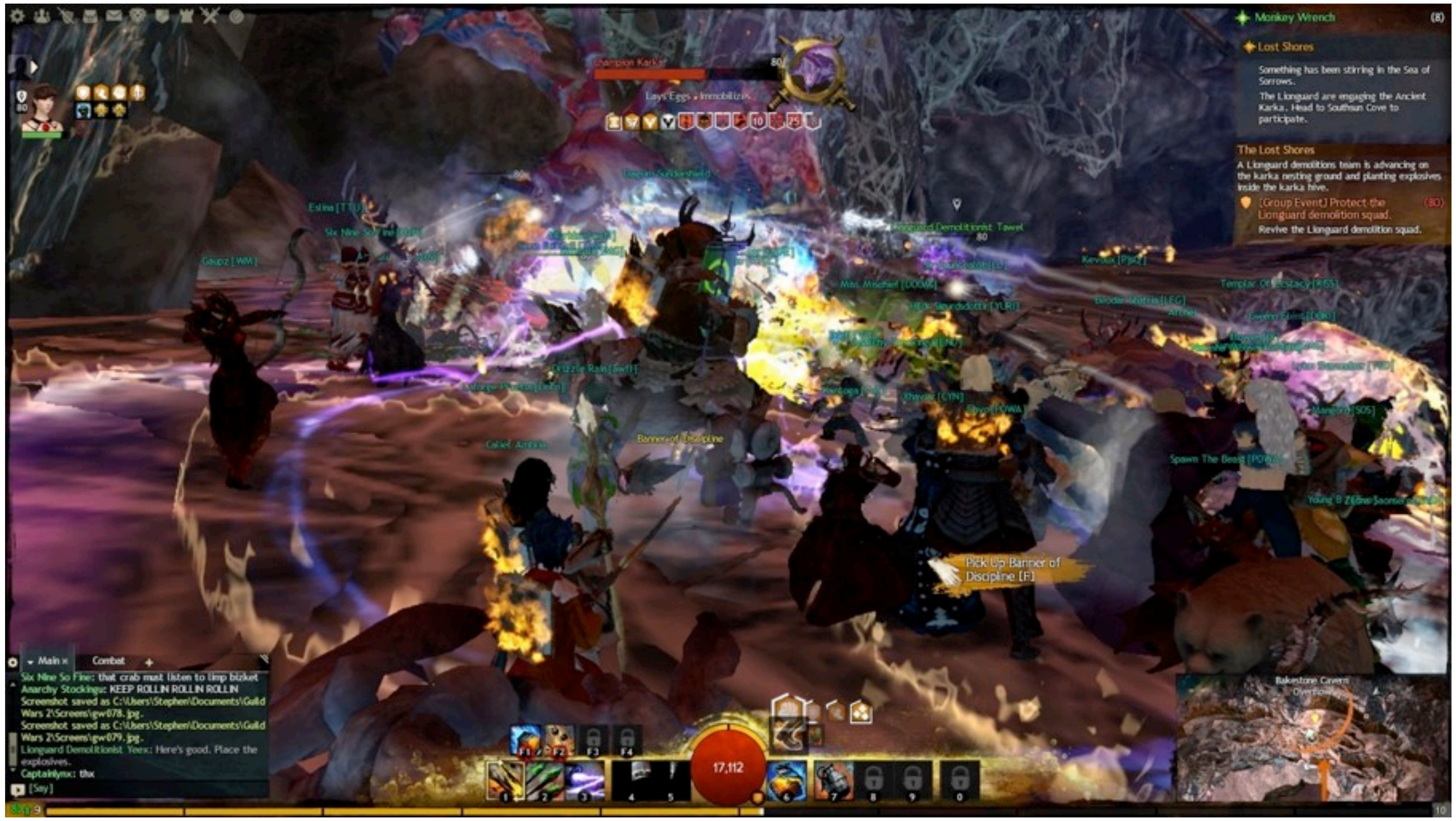
Screenshots

Screenshots

Screenshots

Bad Forecast
Invade Kudu's lab and confront him about the weather control device.
Meet Zojja and Peredur at the Invariant base.
1,541

Screenshots



Monkey Wrench (8)

Lost Shores

Something has been stirring in the Sea of Sorrows.

The Lionguard are engaging the Ancient Karka. Head to Southsun Cove to participate.

The Lost Shores

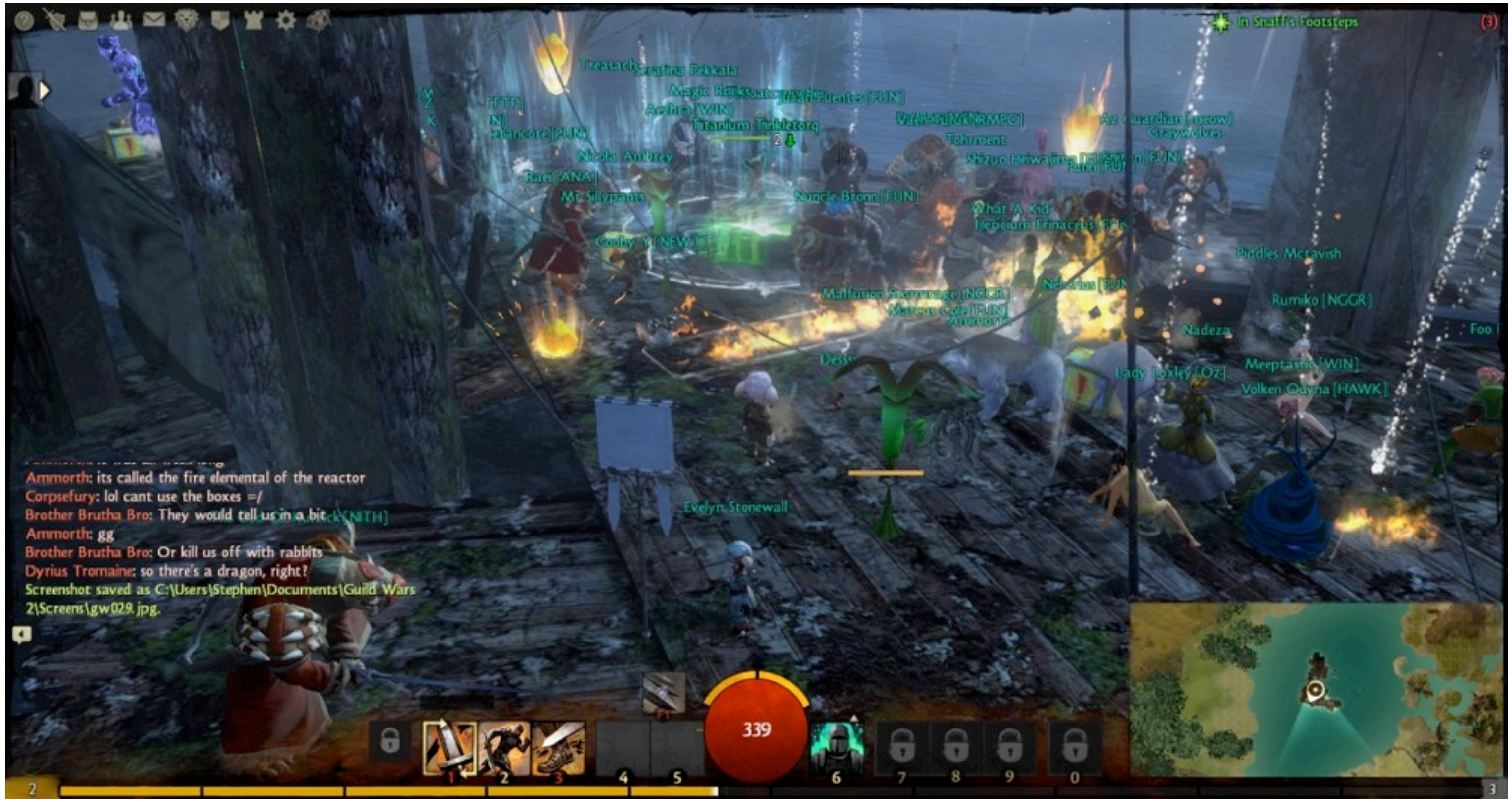
A Lionguard demolitions team is advancing on the karka nesting ground and planting explosives inside the karka hive.

[Group Event] Protect the Lionguard demolition squad. (80)

Revive the Lionguard demolition squad.

Six Nine So Fine: that crab must listen to limp bizket
Anarchy Stockings: KEEP ROLLIN ROLLIN ROLLIN
Screenshot saved as C:\Users\Stephen\Documents\Guild Wars 2\Screens\gw078.jpg.
Screenshot saved as C:\Users\Stephen\Documents\Guild Wars 2\Screens\gw079.jpg.
Lionguard Demolitionist Yeex: Here's good. Place the explosives.
Captainlynx: thx

Screenshots



Ammorth: its called the fire elemental of the reactor
Corpsefury: lol cant use the boxes =/
Brother Brutha Bro: They would tell us in a bit
Ammorth: gg
Brother Brutha Bro: Or kill us off with rabbits
Dyrius Tromaine: so there's a dragon, right?
Screenshot saved as C:\Users\Stephen\Documents\Guild Wars 2\Screens\gw029.jpg.

End of screenshots

Each screenshot was on the big screen for about 1/4 of a second. But you can look at them as long as you like. Or watch videos on YouTube. Or Google even more screenshots.

First User Story

"As founders of ArenaNet we want to create high quality products ... and the best way to do that is through iteration."

-or-

"As founders of ArenaNet we desire agility and iteration ... and we are going to structure our new company around those goals."

Key fact: I am not a founder.

I mention that I'm not a founder, in case someone is confused as to how user stories work. This user story was related to me by Mike O'Brien, one of the founders.

… ergo … the ArenaNet Development Process

- Process built around continuous integration and ***continuous delivery to an online audience***.
- The build process at ArenaNet is in the DNA of the company.
- Artists, designers, programmers, sound wizards, and writers see the results of their work in a dynamically updated online world as it is developed.
- It’s magic!

Continuous delivery to an online audience at huge scale.

- Anyone can push a build on Dev, Stage or Live (although a Live build may not always be a wise thing to do!)
- I wanted to kick a Live build here, right now, from my laptop, but the schedule didn't quite line up for the March release, which came out yesterday.
- ***Secret Sauce: Any version to any version patching.***

Any version to any version patching

- The minimum update required to play the game is computed on an individual basis for every player!

- In Gw1, about 25 file servers around the world crunched, cached, and delivered the diffs.
- In Gw2, a handful of our own servers feed Akamai, which claims 100,000 edge servers spread around the world.

Build Server(s)

- Originally, one build server supported the entire company; the build server for Gw1 has every file ever created, and the current size of it is a mere 20,369,229,326 bytes.
- Gw2 had one build server until recently, and the Gw2 patching source file (after yesterday's build) is 195,676,800,694 bytes.
- Next step: 20 simultaneous 'feature teams'.
- "Build Wars"

"Build Wars" refers to either (1) people fighting over the build server; or (2) points earned for breaking the build. The most points ever earned was by a founder who broke the build 3 times and was convinced his 4th and final submission would work, so he drove off to catch an airplane to Europe. You can guess what happened.

Simultaneous overlapping feature development at scale

•How fast we make stuff: Content 'churn' for a year leading up to release: about a gigabyte of new/changed compressed files *per week.*

●Switched from "one studio making one game" to 20 feature teams making new features and content for Gw2, and delivering new content to players once or twice a month.

•First refactoring of the build process in 12 years (yesterday’s Live build used the new build server - w00t).

•We plan to create build servers on demand by pushing a button.

Summary: Agility at scale

Developers (about 200 peeps)

- 20 Feature Branches (~10 peeps each)
- Alpha Branch
- Staging Branch
- Live Branch

(Live for us too! ... and ... servers.)

Until the day before I gave this talk, we had never played the live build before the players at home! Yesterday we poked around in it for almost an hour! Also, server programmers used to build server code on their local machine and post it live! This allowed for great agility. Most producers faint when they hear this. These days server builds go through the build server, so we always know the source code matches the deployed server. The old way was exciting though.

Next User Story: Iterative programming at scale

"As a founder, I want programmers to be as productive as possible, so that they have fun, and we as a company write tons of awesome code."

My personal dev environment

- Four core, ~~8~~ 16 gigabyte machine, with SSD.
- Guild Wars 2 code base builds in six minutes.
- At runtime the client can use pre-crunched asset files via Akamai (same system used for Dev and Live) or locally stored art and design content.
- Typical server changes compile and run in about 40 seconds. (Client takes 90 seconds to link; feels like forever.)

After I gave the dry-run of this talk at work, one of my colleagues verified that it only takes six minutes to build the servers and client on our dev machines. Sorry, I said 8 gigabytes during the talk, but it turns out I have 16 gigabytes.

Why such fast builds?

- Stock Visual Studio 2008 or 2012.
- Not using IncrediBuild or other build accelerator.
- So ... how?
  - Strict use of interfaces.
  - Strict adherence to minimal include file references.

Running locally

My computer can easily run everything we also run in a datacenter:

- Database(s)
- “Platform”
  - "Login servers" (login, chat, guilds, parties, friends)
  - Caching & Deployment
- Commerce stack
- Game server
- Game client

Remember: the game client wants to use up as much of the machine as possible! Even with it consuming half of the 8 gigabytes (since it is a 32-bit app it is restricted to 4 gigs) the rest of the code - the server stack - runs fine.

Summary of dev environment

- "Programming in the small" - but at scale.
- Generally and correctly speaking, I run the same code locally that runs in the datacenters.
- Most #ifdefery is about enabling cheats in the different branches.

Next User Story: Fun with threading

"As a server programmer, I want to know the ideal number of threads we should allocate when we deploy game servers for Live players."

-and-

"As a server programmer, I want the game servers to automatically use the minimum number of threads to aid in debugging and performance measurement."

Key fact:

- We run hundreds of game maps under a single process; we do this for control over the scheduling of maps and because of ports.
- Generally speaking, the game server ‘process’ is the only thing running on the hardware.
- That’s why we want to know how many threads should go into our single game server

... so ... how many threads?

Goal:

- Game servers should use **as few threads as possible** but **as many as necessary**.
- Why?
    - Using the ‘exactly correct’ number of threads optimizes the utilization of processor cores.
    - But when debugging it's best to run few threads regardless of the number of cores.
    - Want same behavior from the same code on local machine as in datacenter (e.g., for Telemetry).

... in other words, we always want the game servers to use as few threads as necessary. When debugging locally, it’s easier when “game context” (maps) don’t jump threads; and the same is true in the datacenter, when using Telemetry (RAD’s profiling tool) or our own telemetry-style code.

Typical job system ...

Natural solution:

- Keep a list of threads and a priority queue of ready tasks, and dispatch them as they become ready and as threads are available (typical “job system”).
- Drawbacks:
    - No good way to detect blocked threads (blocked on I/O, critical section, whatever) and dispatch another one that could be doing useful work...
    - And start new threads only when a thread is genuinely blocked (otherwise we just create resource contention).

Trick: use Windows I/O Completion Ports (IOCP) because they have magical properties.

- When an I/O event completes Windows will dispatch a thread, if available, to handle the I/O event. (Otherwise the event is queued until a thread is available.)
- Normally Windows will dispatch only as many threads as cores, to prevent thrashing from context switches.
- However, if one or more threads are 'blocked', Windows will allow more threads (if available) to run than the number of cores.
- MSDN and “Windows Internals” says “a thread is interrupted if it blocks for a significant period of time.”

Context switching

- There is no context switching (between threads inside your app) if the number of threads == the number of cores.
- Conversely, in the same configuration, if all threads 'block' at the same time, there are no threads that can take up the slack.
- If we create too many threads we waste resources and also increase the odds of an annoying context switch.
- *Cost of context switches is NOT THE OS.*

On modern hardware, the OS switches between threads at an astounding rate. However, switching between compute-bound threads is expensive, because switching to another thread will generally flush the cache of the previous thread. That’s not good.

... so ... (again) how many threads?

- You will see the following statement over and over again in books and and articles (this exact line from MSDN):

  "... a good rule of thumb is to have a minimum of twice as many threads in the thread pool as there are processors on the system."

- Key fact: it is only a rule of thumb - and not a very good one.

- Only proper solution is to measure.

... so ... how many did you choose?

a. Number of cores * 2 (as per tradition)
b. Number of cores + 2
c. One
d. 200
e. Dynamically allocated based on demand.
f. Number of cores - 1

For Gw2 game servers, the answer is ...

The answer is ‘b’

For game servers only:
b. Compute threads: **# of cores + 2**

(Plus a few other threads for file and socket I/O.)

The answers “One” and “200” are also excellent answers for other parts of our system.

"Number of cores + 2" ... ?

- In order to measure how often a thread is getting blocked, you need more threads than cores, so Windows will dispatch an extra thread if blocking occurs.

- Number of cores ...
  - +1 The "compute thread pump" which dispatches an extra thread 25 times a second (which often quickly returns [analogous to I/O notifications dispatching threads]).
  - +1 An extra thread that should only execute if all other

This is basically using IO Completion Ports without IO! Instead a notification is posted to the IOCP which wakes up an additional thread every 1/25 of a second. That thread looks around to see if a game context should be run; if not, it quickly exits.

Brief aside: I/O Contention

"As a server programmer, I want to minimize contention between compute threads and I/O threads."

- We also have I/O threads (otherwise nothing would happen).
- The original Gw1 I/O threading code required at least two.
- After measurement, I sorely wanted to reduce this to just one, but ran out of time to ensure there would never be a deadlock.
- These threads (on game servers) simply queue requests and exit.

I skipped this slide in the actual talk. If I had not skipped it, I would have said that our I/O threads act like Windows Deferred Procedure Call (DPC) handlers; the get completion notifications and hand off the actual work to another thread as soon as possible.

Next User Story: Measuring compute thread blocking

“As a server programmer, I want to know how often our compute threads actually block, so that I can choose the ‘right’ number of threads to allocate.”

I love Stress Tests

- The only way to find out how many threads we would really need was to run with Live users, ideally ***before*** the first Beta Weekend Event. (We didn't have the final hardware configuration during CBT3.)
- Since Beta Weekend Events are meant to show off the game in the best possible light, it's not a great time for experimenting!
- I doubt we invented this kind of "stress testing" between Beta events, but we sure pushed the envelope.

Stress tests FTW

We had 12 stress tests - 1 private (with the CBT crowd), 10 public, and 1 accidental. Contrast with 3 BWEs.

First World Problems
GUILD WARS 2 IS HAVING ANOTHER STRESS TEST.
BUT IT'S ONLY ONE HOUR LONG.
quickmeme.com

Tuesday, April 2, 13

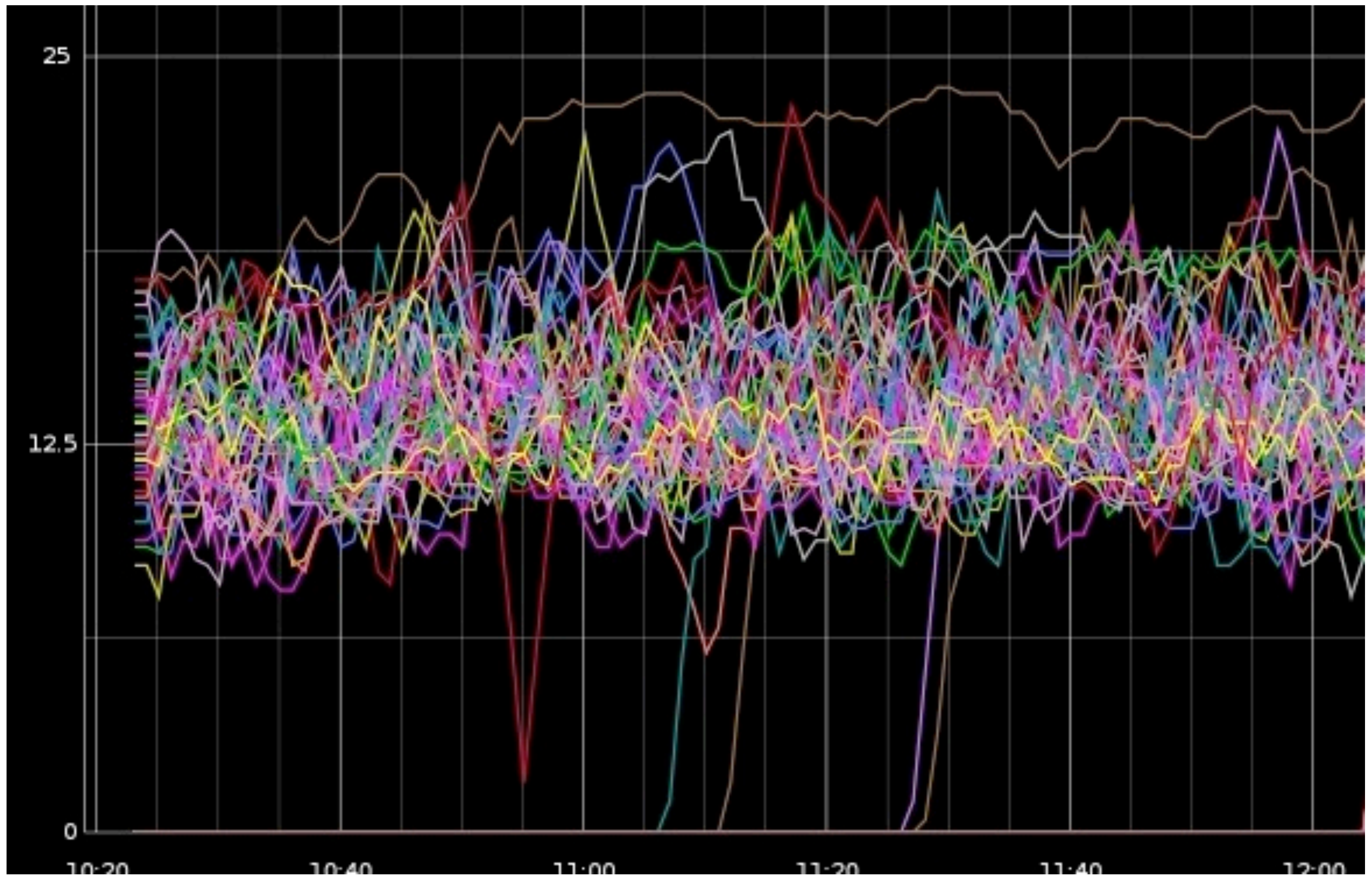
The accidental one was caused when we updated the Auth servers and a few minutes later the Community team said, “You guys know people are logging in and playing, right?” “Whazzat”, we said? Rather than just kill everyone, we fooled with the Auth code “old-school style” with manual server restarts. It was fun.

Stress Test Thread Concurrency

25

18

12

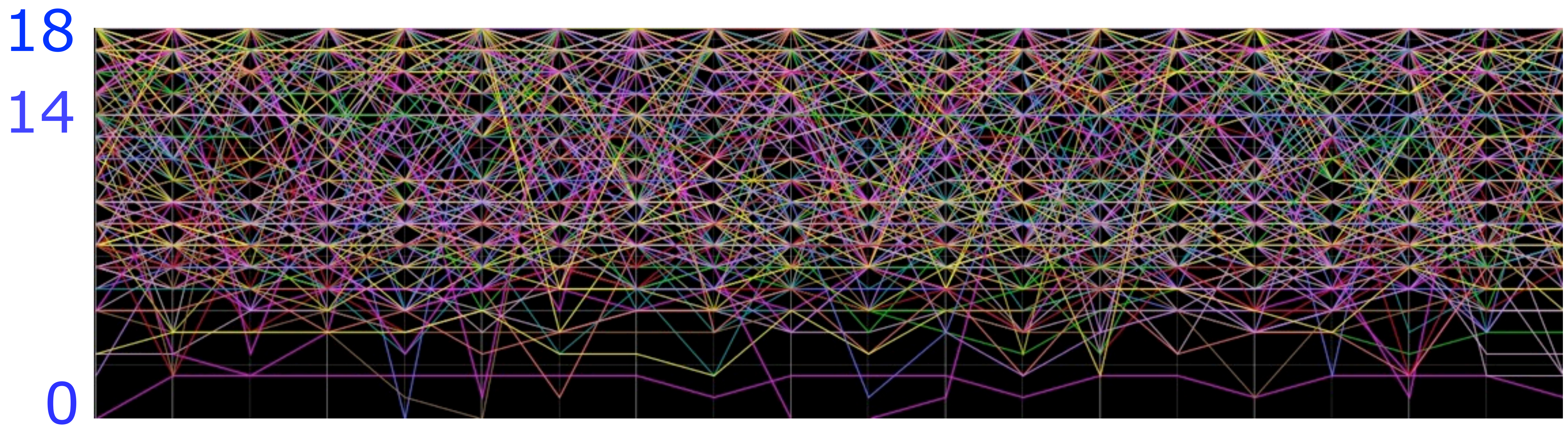


Threads in use over time

During our first stress test and using our fresh installed Graphite visualization tool we monitored the number of threads in concurrent use. You can see it clusters around 12 threads (which matches our number of cores) and goes as high as 18 threads (ignoring the outliers – who cares about them). We had to decide on a number between 12 and 18.

Thread Concurrency Final BWE

WTF is this chart anyway? It is the minute-by-minute number of threads in use across dozens of hardware servers.



We have dispatched as many as 400 maps/server with this.

This is the number of concurrent threads in use sampled once a minute, with each server assigned its own colored line. With practice you can trace the number in use by following the lines. You can see a horizontal line at 14 and then of course the cap at 18 (we set the cap at 18 for the final BWE). We decided we hated contention more than we liked concurrency so we choose 14, as with that number we could monitor blocking and minimize contention.

Key fact:

- Our "sweet spot" for server hardware is 12-core blades, and "*one thread / core + two"* has worked well for us.

    •We tested an Intel 32-core machine and an AMD 48-core machine.

    •In both cases, simply due to the massive game maps we run, our choice would have been to run with fewer threads than cores! Because eventually memory contention became an issue for us with that many cores. ***YMMV***.

Our experience is that 3 12 core machines are much, much better than one 32-core machine from the same vendor. Again, YMMV, but with 3 12 core machines, not only do you get 4 extra cores (total 36) but a massive increase in memory bandwidth, because each set of 12 cores has (effectively) dedicated DDR3 memory.

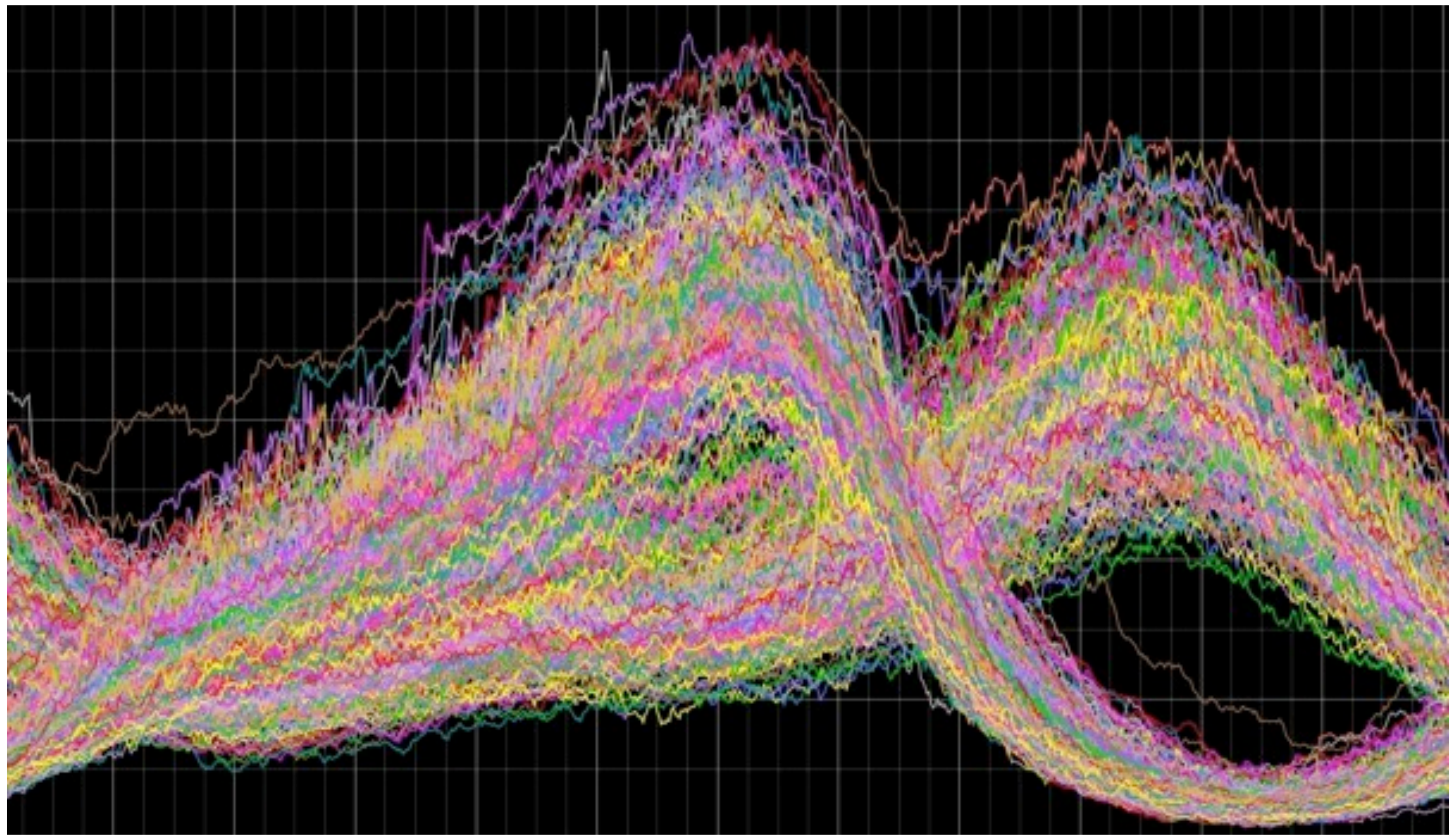
Next User Story: Horizontal Scaling ...

“As a server programmer, I want to allow as many people who want to play, to play without login queues.”

- Thread efficiency is one thing; raw compute power is another thing. We load balance thousands of maps (public maps plus story instance and dungeon maps) across hundreds of hardware servers in each datacenter.
- Instead of login queues, we have overflow maps, which allow players to keep playing while they wait for their home instance.

The Dragon Graph

Concurrency per blade; both datacenters



24 hours

Per blade concurrency: NA & EU

EU: 5:1 range; NA 2.5:1 range

Someone at NCWest called this the dragon graph and it stuck. This is the player concurrency on each hardware server in both datacenters over a 24-hour period. You can see that NA and EU have different peak concurrency times. When we quote concurrency, we are quoting world-wide simultaneous use of the game. I think a reasonable alternative would be to add up the peak per datacenter, which would be higher, but that's not what we do.

Next User Story: Server populations

“As a designer, I want ‘worlds’, which support social interaction, and allow us to create ‘World vs. World’ which is easy-access PvP.”

“As a server programmer, I don’t want to hardwire server capacities so that we have maximum load balancing flexibility.”

... so to answer everyone’s question authoritatively: the world capacities are managed and set by hand. They are based on the total membership of the world. It can’t be otherwise, because as you can see in the previous slide, the population varies enormously during a 24 hour period. We change the mix of “Full”, “Very High”, “High”, and “Medium” in order to encourage people to balance the worlds. Today, world transfer to “Medium” is free to support our latest WvW update; this encourages people to transfer to less busy worlds, thus balancing things out. At the same time we made transfer to ‘medium’ servers free, we also increased the number of ‘medium’ servers by reconfiguring the world population settings.

Next User Story: Virtual Machines

"As a server programmer, I want to control how game server maps are dispatched."

- So ... we don't use VMs (from any provider) after a bad experience on Gw1.
- VMs are great for many things, just not our game servers.

We understand Windows scheduling and we understand our own thread scheduling and adding in another scheduler from the VM provider just confuses the issue. Specifically, for Gw1, some servers are low-load and low-latency, and the VM software liked to put those to sleep because they weren't very busy. Sometimes for 20-30 seconds. That broke things. We had tested game servers under VMs because we expected that's where the problem would be but (doh!) the problem was really with some of the backend servers! *Sigh*. We had to quickly rebalance our server deployment during the first four hours after the complete conversion to VMs. BTW, we used VMs for Gw1 because it was designed to run on 2 processor (yes, processor, not core) systems, so when we bought all new hardware, we decided to emulate on 12-core machines the old layout. When that didn't work, we made our own system for running multiple instances of our servers on a single piece of hardware and now we only use VMs for certain utility purposes.

Next User Story: Memory shortage masking as CPU hiccup.

"As a game programmer, I don't want the game server to suddenly go out to lunch, so server programmers please figure that out."

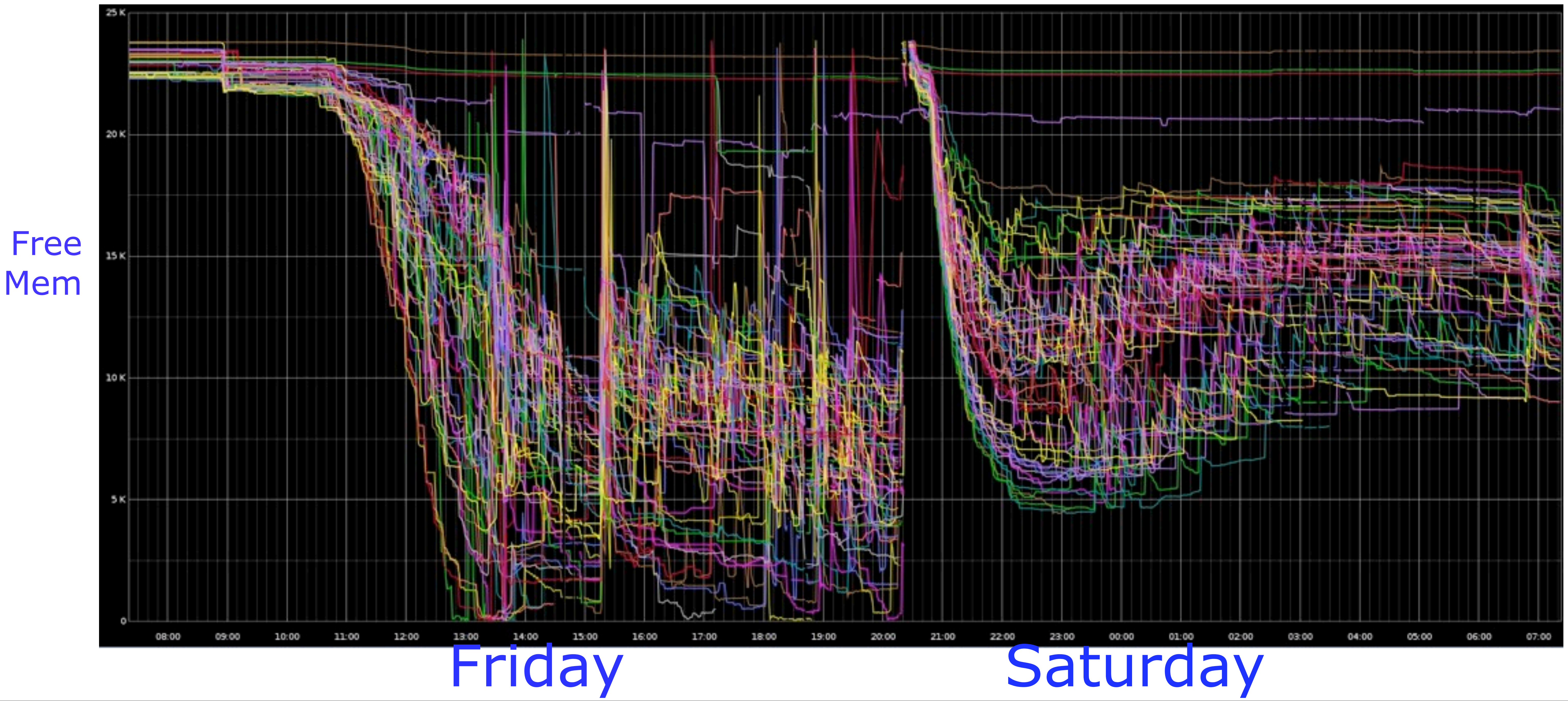
- Tons of time spent on tracking down contention, especially in critical sections.
- But during CBT3 there would be these huge CPU spikes … just sometimes.

Transitioning from 32-bit to 64-bit

- Long story short: when 32-bit apps run out of memory, they crash. When 64-bit apps run out of memory, they page (but you knew that).
- When your server is seriously over-budget, memory-wise, you know it. **When you’re a tiny bit over budget**, and you are used to 32-bit app limitations (where the code will crash when out of memory), you see what looks like a CPU hiccup.

... and it was self-regulating, at least during CBT3 and BWE1: the CPU hiccups are experienced by players as lag (spell activation delay) and so when it is bad, they logout, and when enough logout, the problem goes away. Therefore the game servers bounce along using slightly too much memory from time-to-time. We had spent so much time worrying about Critical Section performance ... and many years running 32-bit servers ... that we forgot that it only takes a little bit of paging to ruin your day.

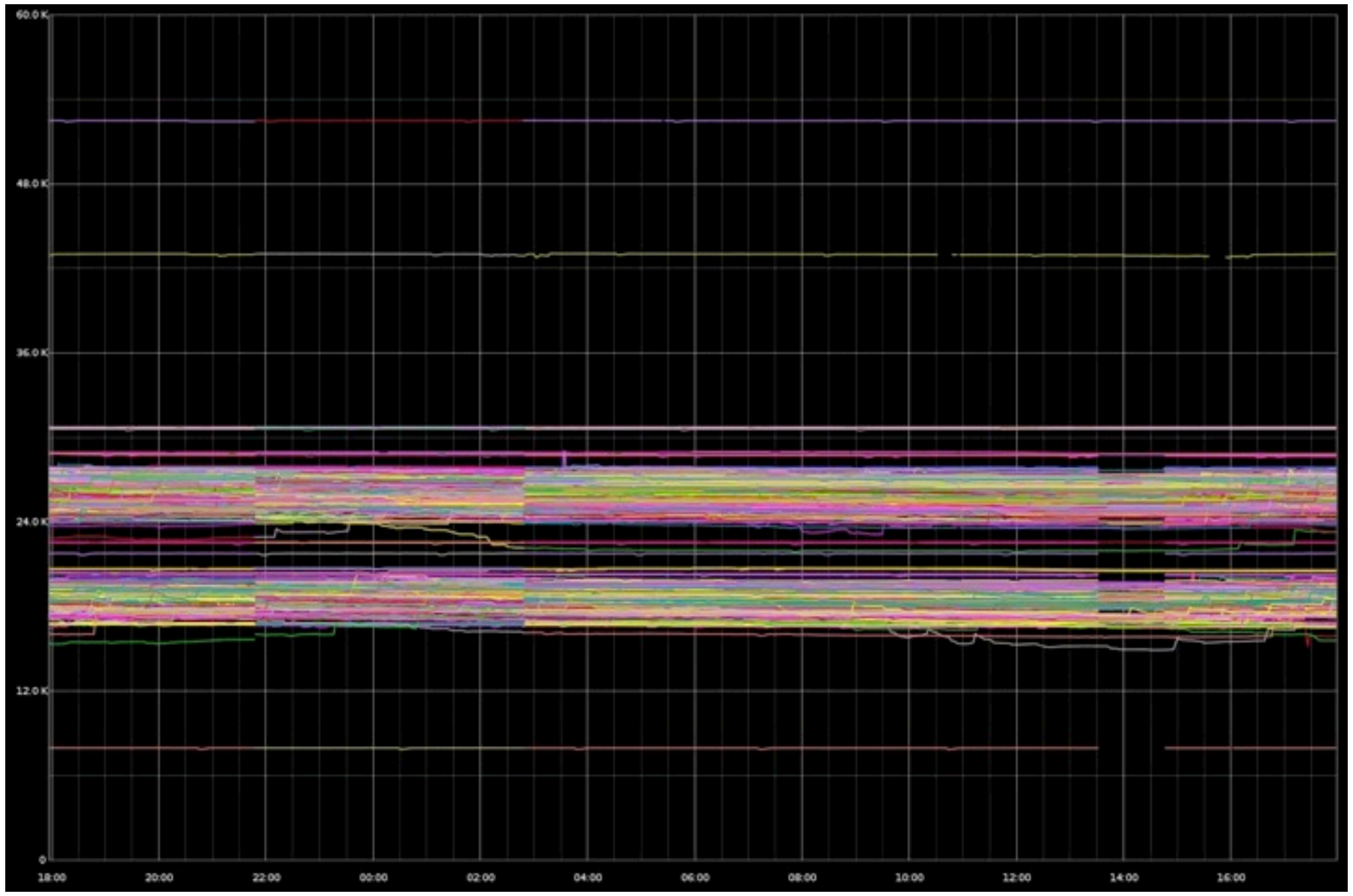
The ‘Waterfall of Death’



During Beta Weekend Events, the concurrency was highest Friday, dropping a bit Saturday, then lowest on Sunday. This chart shows how on Friday some servers ran out of physical memory and started paging; by the second day of BWE1 concurrency was lower and there were no problems. After this, we delayed BWE2 so we could buy more hardware, just in case ... but see next slide.

Fixed for BWE2

Free Mem



Yay! Gameplay programmers figured out where the memory was going – some story instance memory was not being properly reused (but it was reclaimed when the instance exited, so the lack of RAM was never a permanent condition). For BWE2 memory usage was very stable. There are two lines because the new machines had 32 gigabytes of RAM, and the (slightly) older ones have a mere 24 gigabytes of RAM.

Next User Story: Accurate CPU measurement

"As a developer, I want to know how much CPU our game servers are using."

Don't use Windows Perf Counters because the sampling rate is too low. You must instrument your own code.

If you have an app (a web app) or a game like ours that is driven by user input, and you process the user input as fast as possible, then it is highly likely at low concurrency Windows will sample your CPU usage inaccurately, because a lot of the time your app isn't running. Extrapolating from Windows Perf counters in that situation will be deadly.

Next User Story: Guesting cross-Atlantic

“As a player, I want to ‘Guest’ in other worlds, even if they are across the Atlantic from my home world.”

Me: “It’s good to want things.”

Sorry, the trans-Atlantic cable connection just isn’t reliable enough. We’re not wimping out either; solutions like caching your character record in the local datacenter and then copying it across after an outage would be mind boggling confusing for players and really hard to explain. If we add this feature, it will be for GMs first, because who cares about their character records. I want this to work too, but it’s just not practical right now. BTW, world transfer works, because everything is copied to the other datacenter; you might get disconnected when playing cross-territory but that won’t matter to our servers, where all of the other traffic would be local to the datacenter. Also, some less critical features are available cross-datacenter: in-game email and chat and guild membership.

Next User Story: Availability of Guesting

"As a player, I wanted Guesting; you said we could have it sooner, so we could play with our friends on other worlds."

Me: "my bad."

I told a story here how I totally misled everyone at work with how simple I expected Guesting to be to implement, because I had finished all the back-end work. We kept putting a couple of days into it during each 4 week sprint and then QA would find new UI issues. Finally after three months we noticed a pattern – we had no idea of the scope of what we were doing, particularly regarding the effect of Guesting on the UI. We got all the interested parties in a room and hashed out the true requirements. Implementing those finally got us where we needed to be. This is an example of how a project can go bad a few days at a time or 'optimistic programmer syndrome' in action.

Next User Story: Jobs!

"As a programmer, I want to work at ArenaNet."

Visit our jobs booth!

Visit our newly updated page:

http://arena.net/jobs

Summary

- Guild Wars 2 is insanely huge and frequently updated: we have been posting about 500 megabytes of new content (after delta patching) each month.
- We solve interesting problems that only arise at this scale.
- In spite of the scale, we try to ‘keep it small’ and allow fast iteration.

Thank you for listening.

Questions?

(Also ... puppies!)

"... wait, wasn't there four of us in here a few moments ago?"