SCALABILITY

- Hodicska Gergely
  - Web Engineering Manager as Ustream
- email: felho@ustream.tv
- twitter: @felhobacsi

DEFINING SCALABILITY

- It is not:
  - Performance
    - Easier to scale
  - HA
- It is the ability to handle growing amount of work in a capable manner
- Not just technology but people and process

SCALABILITY TYPES

- Vertical
  - Bigger
  - Typically more expensive
  - Sometimes feasible (DB – SSD)
- Horizontal
  - More
  - Typically we need this

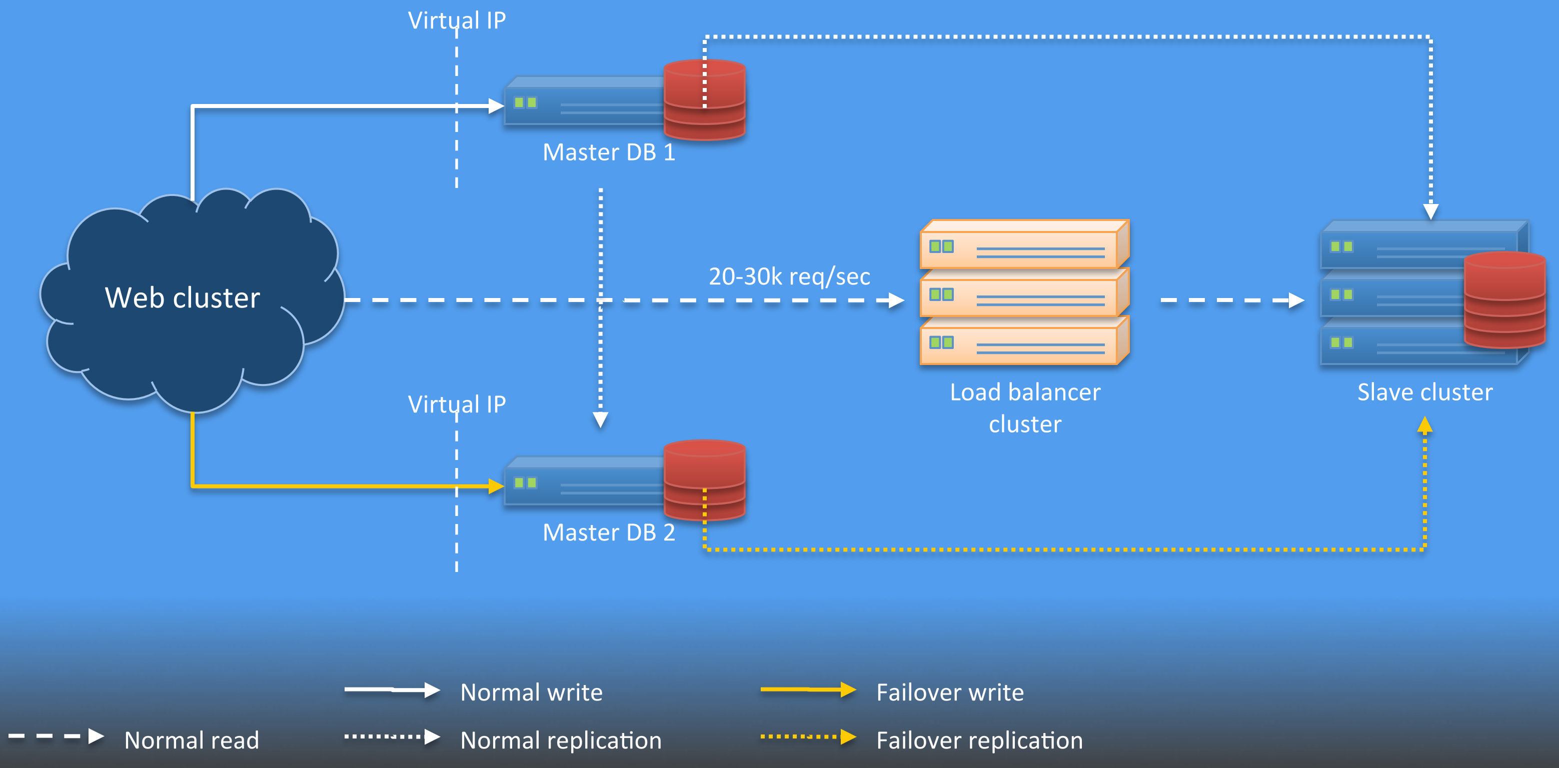
SCALABILITY RULES

- KISS
- Command and conquer
- Approximate correctness
- Shared nothing

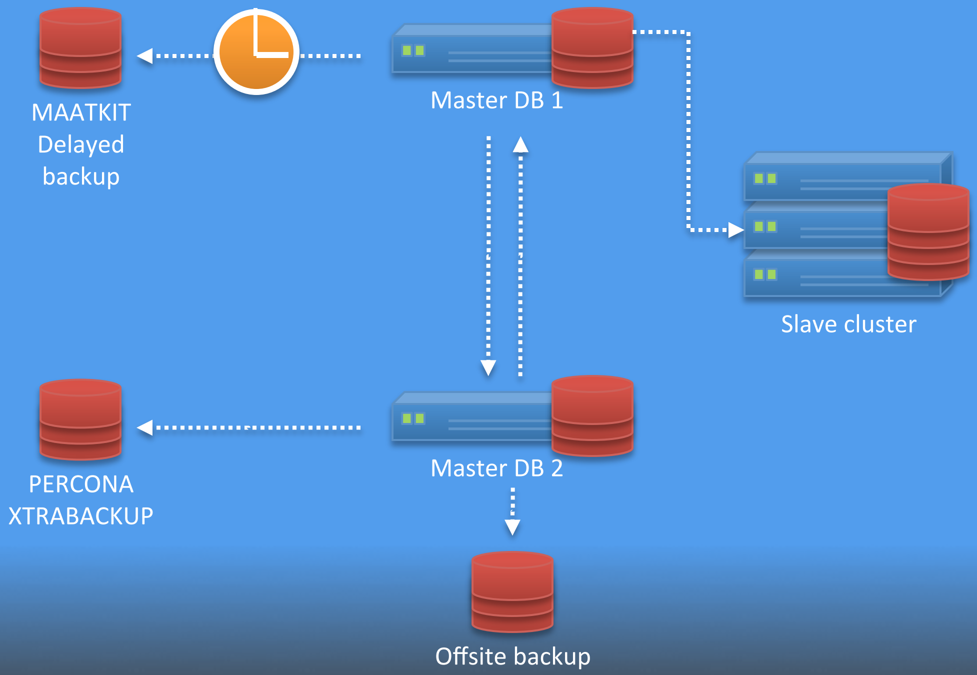
DATABASE

- Most tough to scale
- Read -> Replication
  - Lag, cascading
- Write -> Sharding
  - App logic, vertical vs. horizontal, key server
- HA: Multi Master Replication
  - DRDB, MMM, MySQL cluster

DATABASE – MULTI MASTER MYSQL



DATABASE – BACKUP STRATEGY



- Continuous backup
- Encrypted off site backup
- Delayed replica

NOSQL

- CAP theorem (availability, consistency, partition tolerance -> eventual consistency)
- Diverse world
- Automatic partitioning, sharding, elasticity
- Transparent for the application
- Extendable without downtime
- Fault tolerant
- Redis, Riak, Voldemort, Cassandra, CouchBase

CACHING

- Strategies
  - Write-through cache
  - Write-back cache
  - Implicit/explicit invalidation
- Consistent hashing
- Restart

MEMCACHE

- Local vs. remote
- LRU
- Storing lists
- Versioning
- Race condition (cas)
- Object size

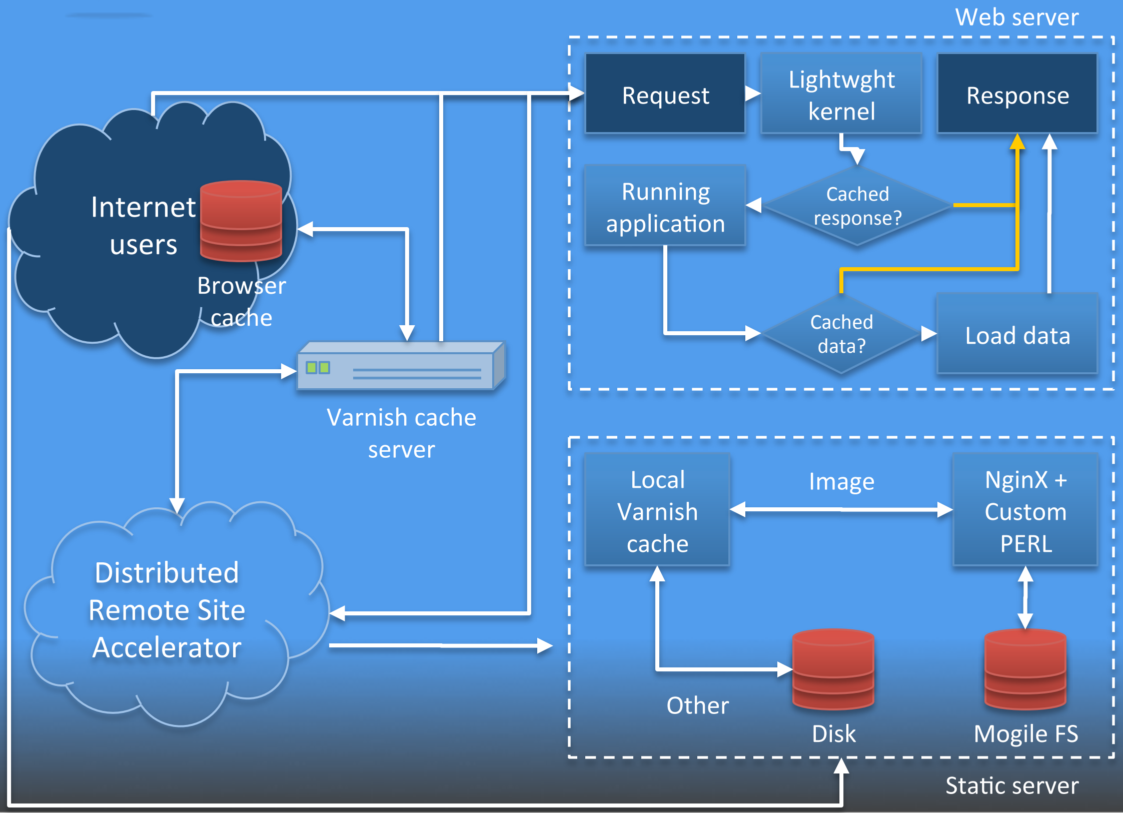
APP LEVEL CACHE

- Code
- Shared memory
- APC, Ehcache
- (Opcode cache)

HTTP LEVEL CACHING

- Cache-Control header
  - Local vs. proxy
- Static versioning
- Huge expire time

CACHING



Highlights

- DRSA: globally distributed reverse proxy to serve the content to the users from a geographically close server
- Cache servers: we try to cache as much requests on these servers as possible to offload our web cluster
- Browser cache: all of our static assets are automatically versioned for optimal serving (using huge expire times)
- Application level: caching whole pages to avoid running all the code or the pieces of data to offload the database
- Our framework automatically package and compress the JS and CSS files to reduce the number of HTTP requests

STATIC CONTENT

- NFS (mount problems)
- DB
- MogileFS
- Authentication, access control
  - Perlbal
- FS limitations
- Low hit ratio

LOAD BALANCING

- Dedicated hardware vs. software based
  - Price
- HA proxy
- Nginx (HTTPS termination)
- LVS (direct routing)
- DNS loadbalance, Anycast (multi site)
- Layer 4 vs. 7

SESSION

- Sticky (shared)
- Centralized
  - DB / NoSQL
  - Memcached
- Cookie
  - Sysop will like you

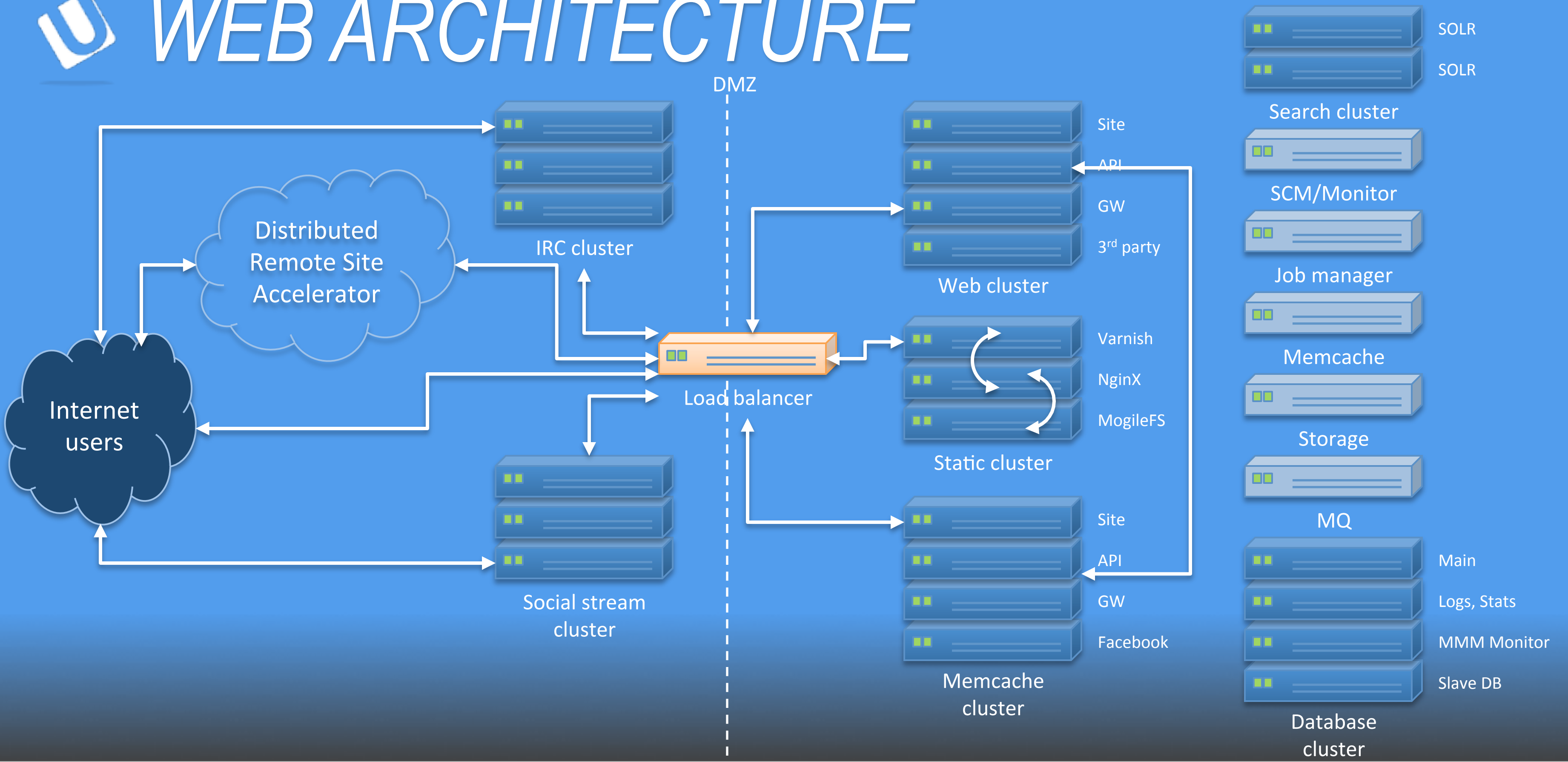
ASYNCHRONOUS OPERATIONS

- Decoupling
- Capacity handling
- Node.js
- Jobs
  - Gearman
- Message queue
  - Q4M, RabbitMQ, ZeroMQ

CHALLENGES OF THE WEB STACK

- Web requests:
  - 200 requests / sec / server (peak)
- Cache server requests:
  - 10,000 requests / sec / server (peak)
- Social stream requests:
  - 15,000 requests / sec / server (peak)
- Database requests:
  - 25,000 requests / sec / server (peak)

WEB ARCHITECTURE

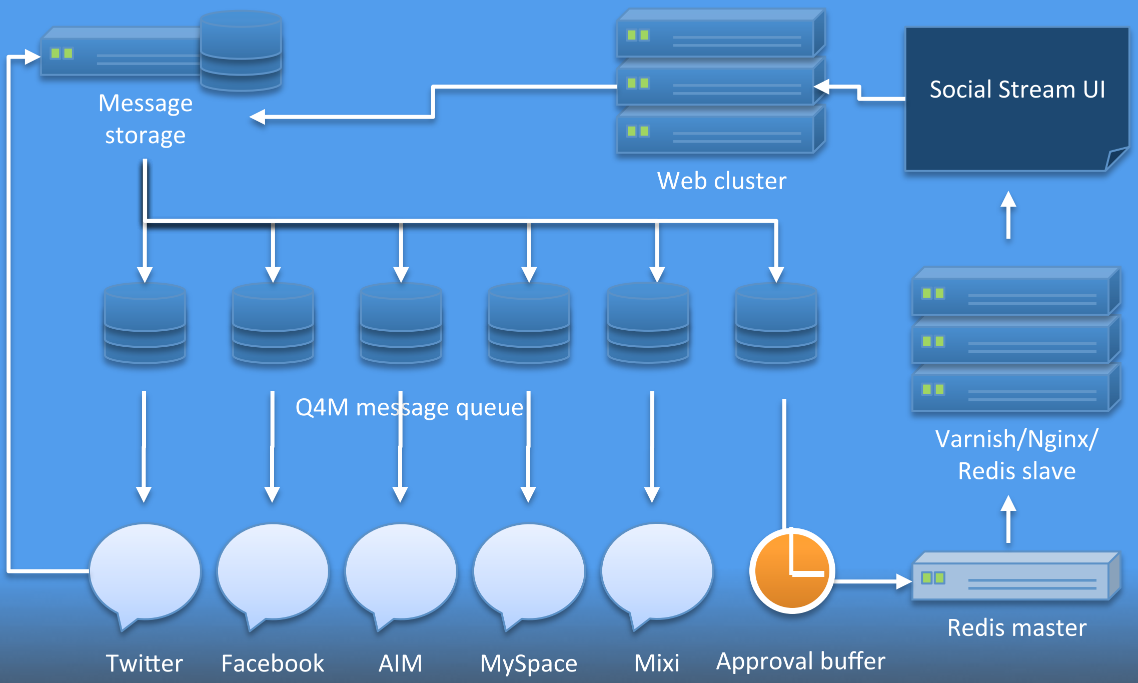


NOW PLAYING

SCALABILITY – BME

19 31 *May 7, 2012* *LIVE*

SOCIAL STREAM: UNDER THE HOOD



Highlights

- Generated 2.5M visits in the last 30 days (the 20% of this is new visitor)
- 0.8-1.1M messages per day
- Justin Bieber has ~230k messages per day
- Jonas Brothers concert: 10k messages per minute in peak
- Peak:
  - 5k new connection / sec
  - 15k requests / sec
  - 600 Mbit / sec

DEVELOPMENT BEST PRACTICES

- Continuous integration
  - Automated builds
  - Unit tests
  - Acceptance test
- TDD, (ADD, FDD ;))
- Abstract branching (feature switch)
- Code review, pair programming, topic experts
- DevOps culture

DEVOPS TOOLING

- Provisioning
- Configuration management (cfengine, chef, puppet)
- Application deployment (capistrano, fabric)
- Orchestrator (mcollective)
- Monitoring (system/application level)
  - Nagios, Munin, Cacti, Graphite etc.
- Supervisors (monit, god)
- Log management/analysis

DEVELOPMENT BEST PRACTICES

- Visualizing
  - Graphite (StatsD, logster)
  - Custom dashboards with KPIs, alerting
- Runtime vs. build time
- Automated code deployment
- Load testing (automated better)
- MVC

CONFIGURATION MANAGEMENT

- Automation
- Versioning
- Accountability
- Chef, Puppet
  - – Same local dev environment

CLOUD ENVIRONMENTS

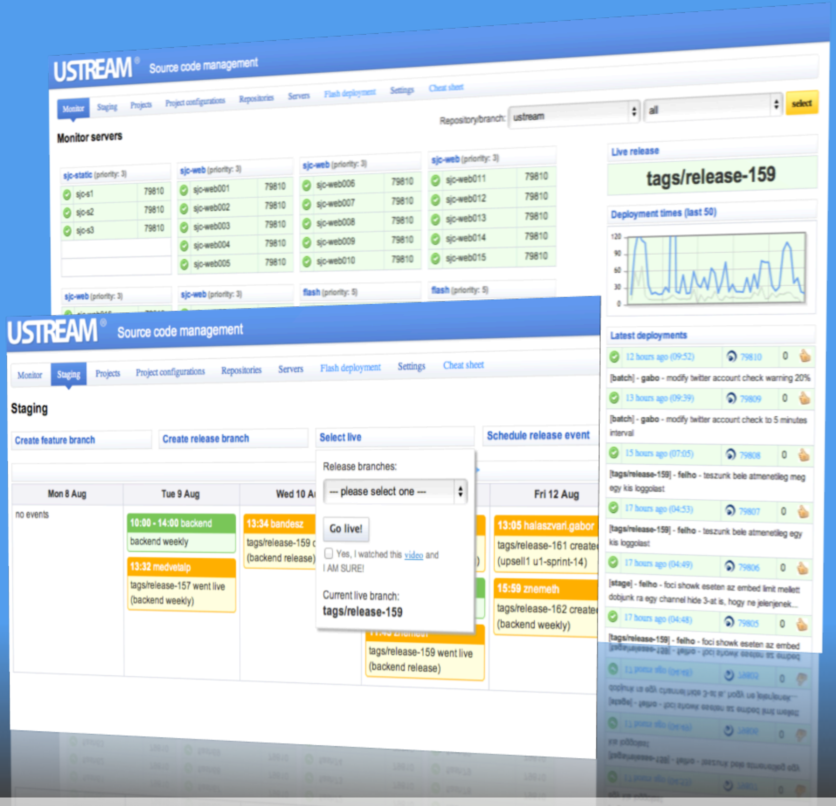
- Amazon, Google App Engine etc.
- Early stage
- Cost savings
- Backup for peaks

ORGANIZATIONAL BEST PRACTICES

- Architect board
- Scrum of scrum
- Product board
- WESK (who else should know)
- Internal demos

SOURCE CODE MANAGEMENT

- Using git for most of the projects
- Code is deployed to more than 120 servers (parallel deployment)
- Custom SCM tool which automates the process
- Different environments: development, staging, releases, live
- Devops culture: developers, sysops and QA work closely together
- Release and rollback policy


USTREAM® Source code management

Monitor | Staging | Projects | Project configurations | Repositories | Servers | Flash deployment | Settings | Cheat sheet

Repository/branch: ustream | all | select

Monitor servers

sjc-static (priority: 3)	
sjc-s1	79810
sjc-s2	79810
sjc-s3	79810

sjc-web (priority: 3)	
sjc-web001	79810
sjc-web002	79810
sjc-web003	79810
sjc-web004	79810
sjc-web005	79810

sjc-web (priority: 3)	
sjc-web006	79810
sjc-web007	79810
sjc-web008	79810
sjc-web009	79810
sjc-web010	79810

sjc-web (priority: 3)	
sjc-web011	79810
sjc-web012	79810
sjc-web013	79810
sjc-web014	79810
sjc-web015	79810

sjc-web (priority: 3) | sjc-web (priority: 3) | flash (priority: 5) | flash (priority: 5)

Live release

tags/release-159

Deployment times (last 50)

Latest deployments

12 hours ago (09:52)	79810	0	
[batch] - gabo - modify twitter account check warning 20%			
13 hours ago (09:39)	79809	0	
[batch] - gabo - modify twitter account check to 5 minutes interval			
15 hours ago (07:05)	79808	0	
[tags/release-159] - felho - teszunk bele atmenetileg meg egy kis loggolast			
17 hours ago (04:53)	79807	0	
[tags/release-159] - felho - teszunk bele atmenetileg egy kis loggolast			
17 hours ago (04:49)	79806	0	
[stage] - felho - foci showk eseten az embed limit mellett dobjunk ra egy channel hide 3-at is, hogy ne jelenjenek...			
17 hours ago (04:48)	79805	0	
[tags/release-159] - felho - foci showk eseten az embed			

USTREAM® Source code management

Monitor | Staging | Projects | Project configurations | Repositories | Servers | Flash deployment | Settings | Cheat sheet

Staging

Create feature branch | Create release branch | Select live | Schedule release event

Release branches: --- please select one ---

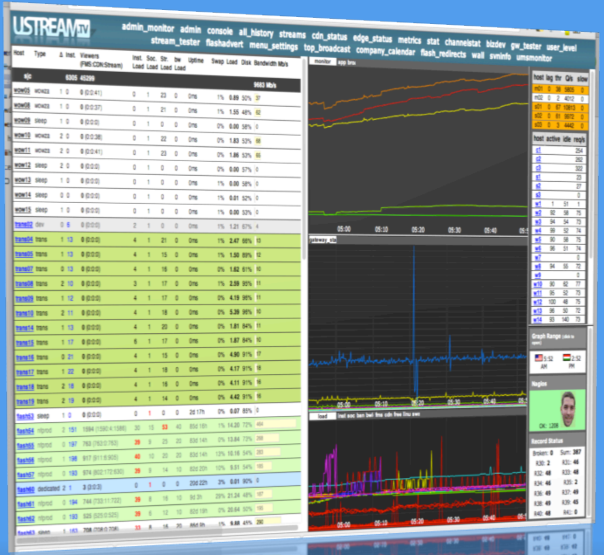
Go live!

Yes, I watched this video and I AM SURE!

Current live branch:
tags/release-159

Mon 8 Aug	Tue 9 Aug	Wed 10 A		Fri 12 Aug
no events	10:00 - 14:00 backend backend weekly	13:34 bandesz tags/release-159 c (backend release)	tags/release-159 went live (backend release)	13:05 halaszvari.gabor tags/release-161 create (upsell1 u1-sprint-14)
	13:32 medvetalp tags/release-157 went live (backend weekly)			15:59 znemeth tags/release-162 create (backend weekly)

MONITORING


USTREAM TV
admin_monitor admin console all_history streams cdn_status edge_status metrics stat channelstat bizdev gw_tester user_level
stream_tester flashadvert menu_settings top_broadcast company_calendar flash_redirects wall svninfo umsmonitor
Nagios

Highlights

- Proprietary dashboard to oversee key system performance charts in one location
- Real-time information about streaming related servers, web/ cache servers, database servers
- Summary of the Nagios checks
- Ability to roll back for historic charts
- Provides shortcut to system tools

Tools

- Munin: Several custom plugins
- Cacti: Mainly for network devices
- Nagios: More than 1200 checks, Active checks
- Monit: Ensuring that a given process runs and it doesn't consume too much resources, Active checks
- Query watchdog: Automatically stops and reports excessive read queries

WHERE TO IMPROVE

- English
- Enjoy programming
- Soft skills (communication, team working, presentation, cooperation, management, leadership, time management etc.)
- Agile development methods (Scrum, Kanban)
- Continuous learning (blogs, books, conferences, code)
- Craftsmanship
  - Clean Coder, Agile Software Development, Martin Fowler books / signature series, The Mythical Man-Month, Code Complete, The Pragmatic Programmer, Peopleware, The Passionate Programmer

WHERE TO IMPROVE

- Network programming, protocols
- Algorithms
- DHT
- Database
- Big Data (Hadoop, HBase, Hive/Pig, BI tools etc.)
- API design (REST, SOAP, oAuth etc.)
- Playing with open source tools of big companies (Twitter, FB, Linkedin)
- Blogging, taking part in open source projects
- Learning different type of programming (e.g. functional)

THANK YOU

Questions?

NOW PLAYING
SCALABILITY – BME

32 31 May 7, 2012 LIVE