Kerberos & HPC Batch systems

Matthieu Hautreux
(CEA/DAM/DIF)
matthieu.hautreux@cea.fr

Outline

- Kerberos authentication
- HPC site environment
- Kerberos & HPC systems
- AUKS
- From HPC site to HPC Grid environment

Kerberos authentication

- Key concepts
  - Trusted third party
    - ☛ Commonly made of 1 server and its backup (KDC)
  - Single Sign-On
    - ☛ Based on Forwardable/Forwarded TGT
  - Limited credentials lifetime
    - ☛ With renewal mechanism
- Footprint
  - Numerous Supported OS
    - ☛ Linux-Based systems, OS X, Microsoft, ...
  - Numerous Supported Services
    - ☛ OpenSSH, LDAP, ...
  - Numerous Supported Distributed File System
    - ☛ OpenAFS, NFS, NFSv4, ...
  - Mostly in private network

Kerberos authentication

- OpenSSH, common usage of kerberos
  - Simplify cascading connections authentication (SSO)
    - Provides connection trees from users to their resources
  - Limited validity through expiration time
    - Each connection associated to a validity countdown, the forwarded TGT lifetime
- OpenSSH, enhanced usage of kerberos
  - Based on cascading credentials refresh
    - Provided by Simon Wilkinson GSSAPI Key-exchange patch
    - Integrated in GSI-SSH (since 4.7)
  - Ease refresh of the connections tree
    - Each connection now associated to the validity countdown of the initial client
    - Initial client credential renew is the single spark to refresh the tree

HPC environment

- HPC key concepts
  - Distributed systems
    - Centralized to be remotely used by numerous users
  - Large systems
    - Thousands of compute nodes/cores
  - Heavy loaded systems
    - From short and small to large and long computations
    - With numerous running and pending jobs
    - With non negligeable delays between jobs submission and start time
- Common HPC components
  - Batch systems and Parallel launchers
    - Schedule jobs, grant resources access and launch computations
    - Slurm, Torque, openSSH, ...
  - Distributed File Systems
    - Share data efficiently between multiple resources
    - Lustre, GPFS, ...

HPC environment

- Common usage
  - Login Nodes connection
    - Using openSSH/GSI-SSH
  - Data staging
    - NAS <-> Cluster FS / Local FS transfers
  - Data processing
    - Application development
    - Results preprocessing/postprocessing
  - Interactive jobs execution
    - With a batch system and a parallel launcher
    - May perform data staging too
    - For application development and validation
    - For pre/post-processing
  - Batch jobs submission
    - With a batch system and a parallel launcher
    - May perform data staging too
    - For non-interactive production computation

Kerberos & HPC systems

- Kerberos interests in HPC
  - Ease user access to compute services
    - ☛ Workstation to login nodes connections
  - Ease compute nodes access
    - ☛ Login nodes to compute nodes connections
    - ☛ For monitoring, debugging, ...
  - Secure data staging stages
    - ☛ Access data on secured NAS seamlessly
    - ☛ For both interactive and batch mode
  - Secure remote connections
    - ☛ Contact external servers securely
    - ☛ For both interactive and batch mode
  - Secured distributed services access
    - ☛ Inside/Outside the clusters
  - Services access tracability

Kerberos & HPC systems

- Kerberos concerns in HPC
  - Credential Lifetime management
    - ☛ What is a common session time in a HPC environment
    - ☛ How to get benefit from kerberos integrated renew mechanism
  - Batch mode
    - ☛ No interactive input from user involved
    - ☛ From where to get a valid credential ?
  - Scalability
    - ☛ Trusted third party behavior with thousands of active nodes
    - ☛ Credential forwarding strategies with thousands of peers
  - HPC specific tools
    - ☛ Are they providing kerberos support ?

AUKS - Description

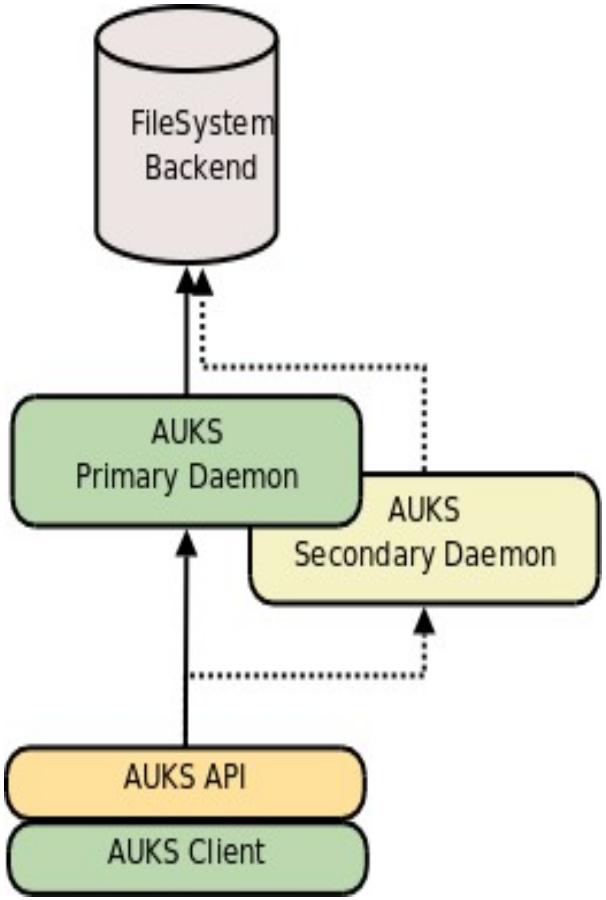
- Goal
  - Provides Kerberos credentials in non interactive environment
    - Batch systems, cron, ...
- Description
  - Kerberos distributed credential delegation system
  - Kerberized client/server application
  - External tool
    - Can be integrated in different projects
  - Linux tool
    - Developed and tested on CentOS, RedHat, Fedora
  - Opensource
    - http://sourceforge.net/projects/auks/

AUKS - Overview

- Internals
  - Multi-threaded C application
  - Based on MIT kerberos implementation only (>1.3)
- Components
  - Central Daemon (auksd)
    - Kerberized server
    - Authorizes requests using client principal and local ACLs
    - Serves add/get/remove/dump TGT requests
    - Stores user TGTs in a FS directory (for persistency)
  - Client API (libauksapi)
    - Kerberized client
    - Provides functions to perform add/get/remove/dump requests
    - Enables third party application to use AUKS functionalities
  - Client program (auks)
    - Encapsulate API functions
    - Enable scripted use

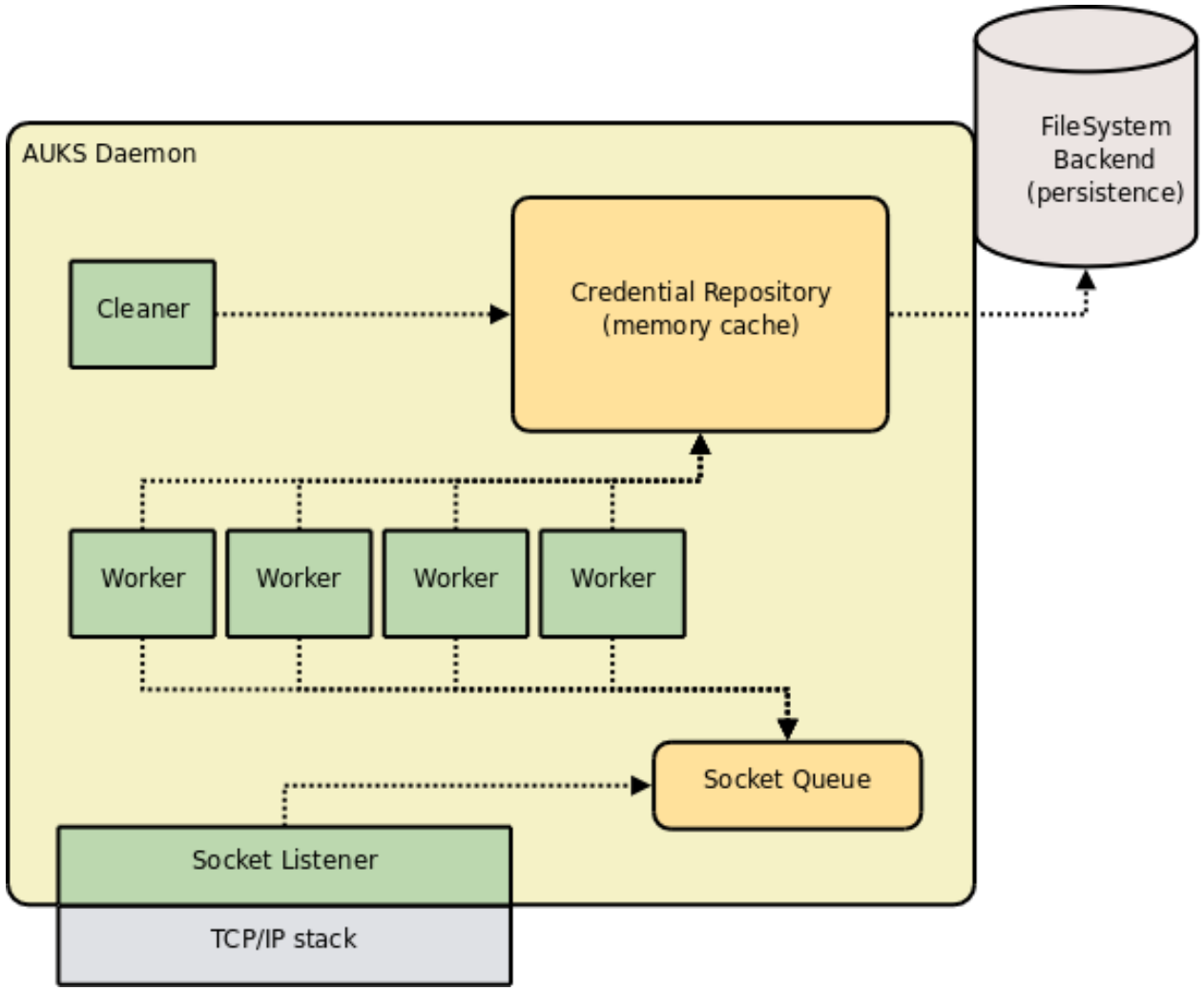
AUKS - Overview

- Auks Features
  - Auksd
    - Stores TGT by uid (TGT principal to local uid conversion)
    - Only one TGT per user
    - Get requests by uid
    - Automatic TGT renew mechanism
  - libauksapi
    - Automatic switch to backup server
    - Configurable retries, timeout and delay between retries
    - Simplify auks integration in external projects
  - HA
    - Active/Passive
    - Rely on external tool (PaceMaker)
    - Requires a shared FS



AUKS - Overview

- Auks Daemon



AUKS - Overview

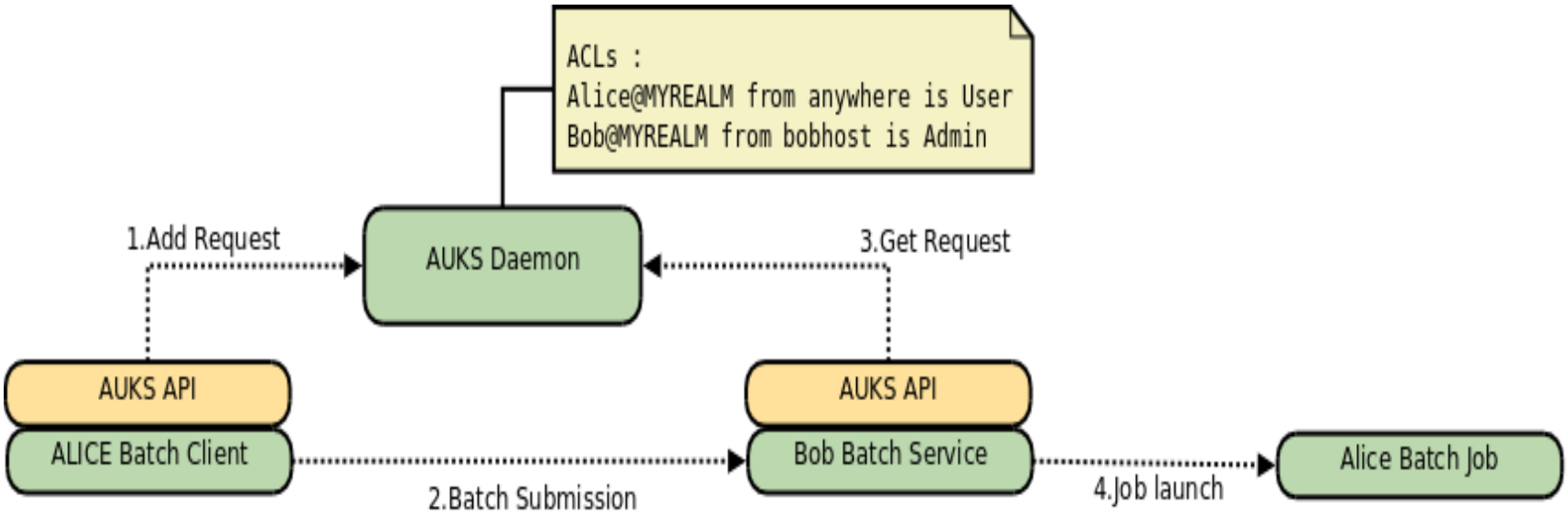
- Auks authorization rules
  - Defined by ACLs
  - Based on
    - Requester Kerberos principal
    - Requester host
  - Determine requesters role
    - Guest : add request for own cred only
    - User : add/get/remove for own cred only
    - Admin : add/get/remove/dump for all creds
- Auks renew mechanism
  - Implemented as a dedicated client
    - Running as a daemon
    - With admin Auks role
    - Dumping credentials periodically and refreshing them when required

AUKS - Overview

- Long running Jobs
  - Users can periodically refresh their Auks TGT
    - Performing a new add request
    - i.e. Once a day, a week, ...
  - Users/Batch systems can renew TGTs using Auks
    - Performing a get request (user/admin only)
    - Automatically using refreshed TGTs
- Scalability in parallel jobs
  - Based on addressless TGT
    - Obtained and used during add request
  - Single addressless credential per user
    - Stored in Auks Memory Cache
    - Provided to requesters without KDC interaction
    - Forwarding to thousands of peers without KDC interaction

AUKS - Overview

- Auks protocol example scenario
  - Alice forwards her TGT to the Auks daemon
  - Alice asks Bob to execute her request
  - Bob asks Auks for Alice TGT
  - Bob executes Alice request using her kerberos identity



AUKS - Scalability

- 3 stages communication protocol
  - Request/Reply/Acknowledgement
  - Leave the TIME_WAIT TCP state on client side
    - Improve server request processing sustained rate
    - TIME_WAIT is 60s long on Linux for ~65k ports
    - Sustained rate > 1100 req/s is not possible
- Replay cache management
  - Enabled by default in kerberos API
    - Uses a single file per user/application
    - Sync file on disk at each addition
    - Multiple threads -> Contention on replay cache
  - Can be disabled on demand in Auks
    - Clusters internal networks can often be considered trusted
    - Greatly improves parallel kerberos communications
    - Choice depending on parallelism requirements

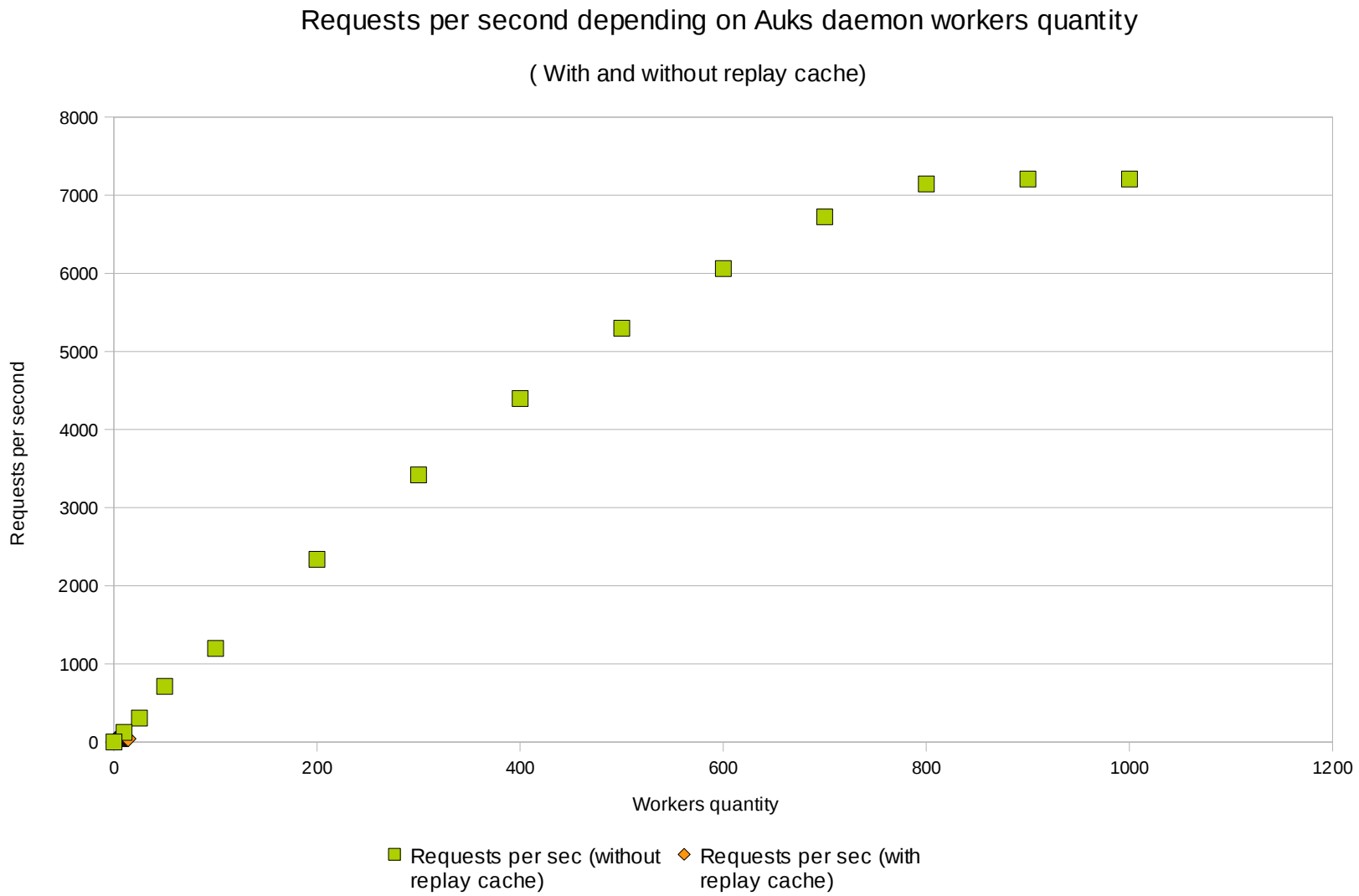
AUKS - Scalability

- Addressless versus Addressed TGTs
  - Addressed tickets
    - Requires a KDC interaction for each forwarding operation
    - KDC is single threaded
    - Auks sustained rate becomes KDC sustained rate (~dozens of TGT per second)
  - Addressless tickets
    - Not need to acquire a new TGT for each requester
    - Sustained rate only limited by Auks internals
- Renew mechanism
  - User/Admin Auks roles enable to get TGTs
  - TGTs can thus be renewed using Auks
    - Renew sustained rate only limited by Auks internals
  - Fallback to default renew mechanism (KDC)
    - In case of temporary Auks failure that would result in invalid credentials

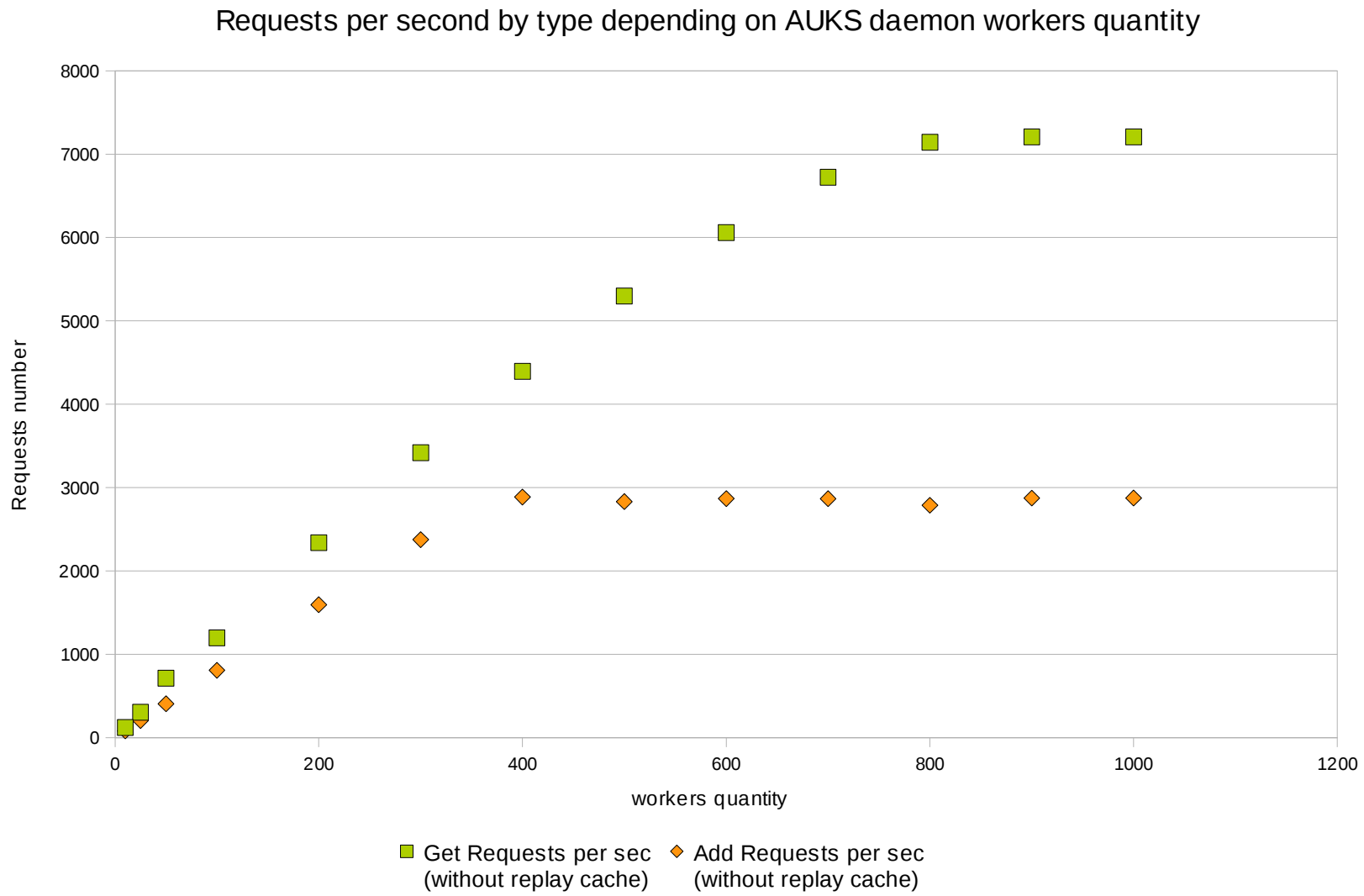
AUKS - Scalability results

- TestBed
  - 1 server + 100 clients
    - SuperMicro 6015TW-INF
    - Bi-Socket Quad-Core (Intel Harpertown 2.8 GHz)
    - 16 Go RAM
    - SATA Intel 3 GBps controller
- Protocol
  - 5 consecutives batchs of 16000 simultaneous requests (20 requests per core)
  - Various quantity of workers
  - With or without replay cache
  - Add versus get requests
  - Measure average number of requests per second

AUKS - Scalability results



AUKS - Scalability results



AUKS – Possible ways of enhancement

- Global scalability by TGS prefetching
  - Current known limitation
    - TGS still acquired using TGT on each node
    - Using basic kerberos API (scalability issue)
  - TGS prefetching
    - Store addressless TGTs and TGSs using Auks Daemon
    - TGS to prefetch based on already acquired TGS and a configurable per principal list
    - As many KDC requests as users multiply by number of different kerberized services + 1
    - Auks becomes a KDC caching system
- Addressed TGT support
  - Better security but with far less scalability
- High-Availability
  - Active-Active architecture

AUKS – Batch systems integration

- Pluggable integration in Slurm
  - A highly scalable resources manager
  - Open source, mainly developped at LLNL
    - https://computing.llnl.gov/linux/slurm/
  - Auks plugin for Slurm
    - Included in Auks tarball
    - Do not provide Kerberos authentication
    - Provide kerberos credential support and renewal
  - Really small overhead in jobs launches
    - Sustained rate up to 7000 req/sec of auksd
    - ~1 seconds overhead for a thousand nodes submission
  - Every user job extends running jobs kerberos lifetime
    - Due to internal Auks refresh mechanism
- Easily integrated in Cron
  - Using auks command line

From HPC site to HPC Grid Environment

- HPC environment
  - Kerberos authentication on workstations
    - With a background renew mechanism
  - GSI-SSH for HPC site remote connections
    - On both workstations and cluster nodes
    - Compiled **without GSI features** (kerberos GSSAPI)
    - Offers cascading credentials refresh (Single point of renewal)
  - NFSv4 + kerberos for remote FS (site centric)
    - Provide NAS with enhanced security
    - Could be replaced with OpenAFS + kerberos

From HPC site to HPC Grid Environment

- Grid environment
  - X509 PKI for user identities management
    - Users own x509 certificates and associated keys
  - GSI-SSH to access HPC sites gateways
    - Compiled **with GSI features** (GSI GSSAPI)
    - Offers cascading proxy certificates refresh (since GSI-SSH-4.8)
  - PAM-PKINIT on HPC sites gateways
    - Experimental pam module to get TGT from proxy certs using PKINIT
    - Linked to GSI-SSH cascading refresh for TGT acquisition
    - http://sourceforge.net/projects/pam-pkinit/
  - GSI-SSH for HPC site remote connections
    - Compiled **without GSI features** (kerberos GSSAPI)
    - Offers cascading credentials refresh
    - Automatically use TGT acquired by PAM-PKINIT
    - Benefit from PAM-PKINIT refresh stages
    - Enables kerberized access to all the HPC site

Questions ?