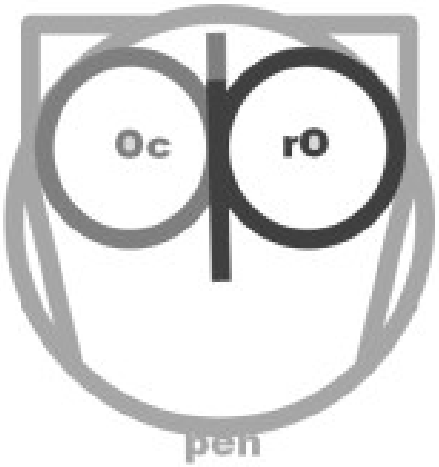
OpenProdoc



Benchmarking the ECM OpenProdoc v 0.8.

Managing more than 200.000 documents/hour in a SOHO installation.

February 2013

Index

- Introduction
- Objectives
- Description of OpenProdoc
- Test Criteria
- Infrastructure
- Realization of the Test
- Results and Graphs
- Analysis and Conclusions
- Using OpenProdoc

Introduction

- OpenProdoc is an Open Source ECM Document Manager developed in Java.
- This document describes different Load Tests performed over several days with OpenProdoc v 0.8.
- The objective of the tests was to measure the behavior of OpenProdoc in different scenarios and to measure the speed that can be obtained with a small infrastructure that can be used even by the smallest organization.
- The tests were run using a program developed for the tests. It's known the expression "Lies, big lies and statistics", so the source and a "test kit" are available with the version 0.8 program. With the kit it's possible to run the test in your environ and compare the results.

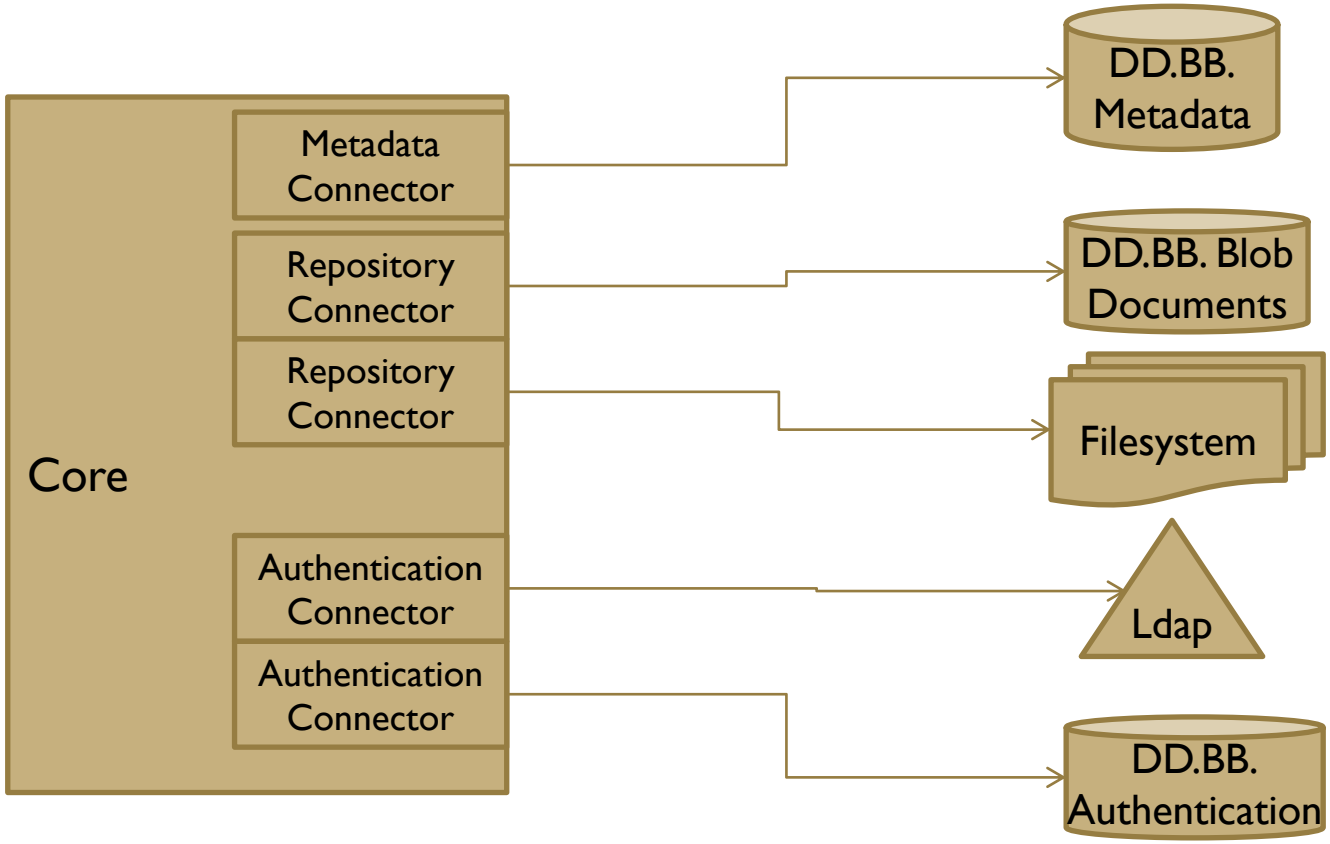
Objectives

- The tests were defined with three objectives:
  - To check the behaviour of OpenProdoc under stress conditions, detecting potential problems.
  - To measure the speed to manage documents
  - To detect the influence of different factors in the performance of OpenProdoc.
- It wasn't objective of the tests finding the database nor the O.S. that works best with OpenProdoc.
- There are too many elements that can affect the behavior of any product, so it was elected to limit to a manageable number of significant factors.
- For automating the test was developed a program that allows to execute different sets of tests and measure results.

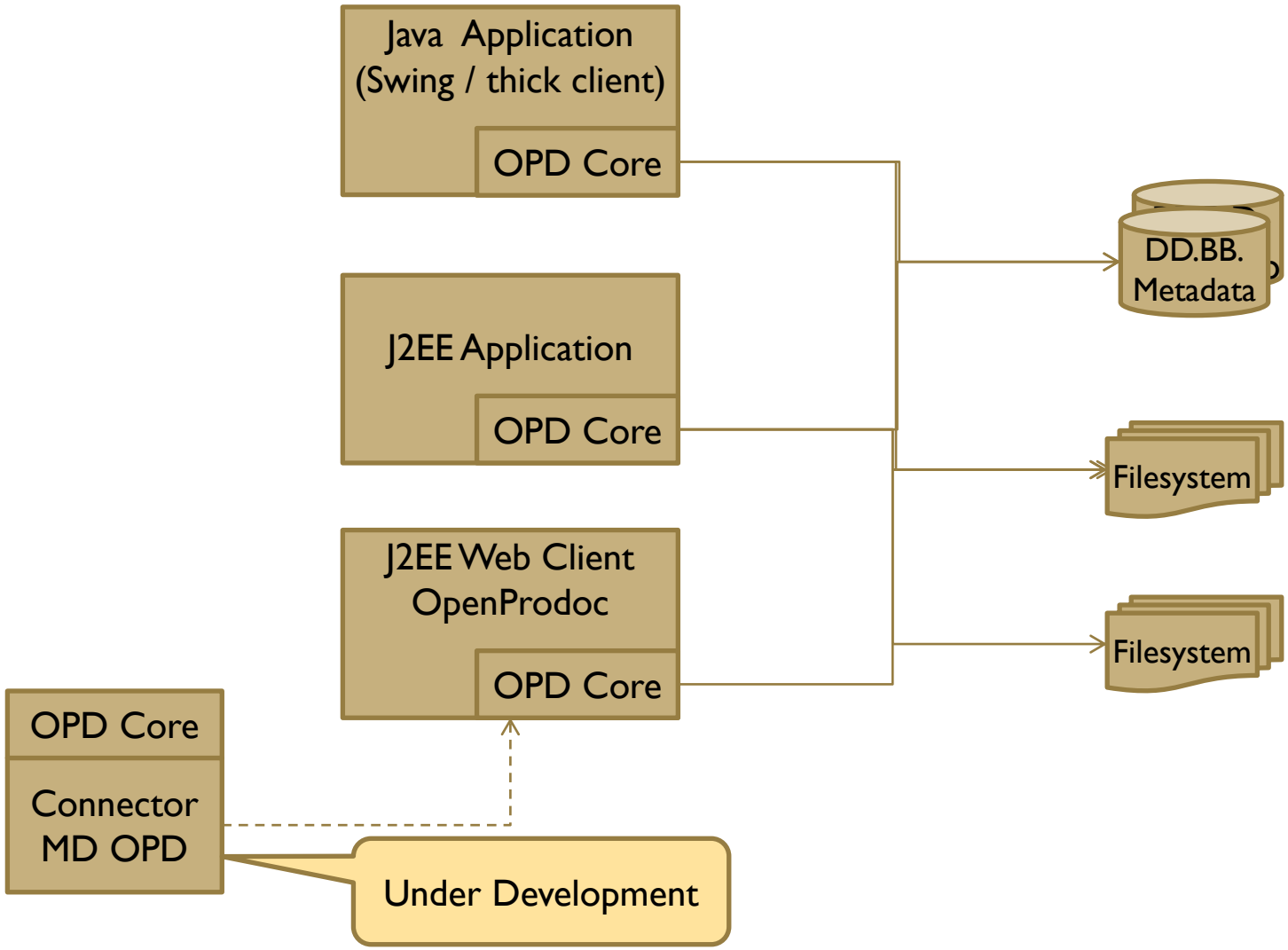
Description of OpenProdoc (I)

- To better understand the elements to be measured and the influence of different factors, it is first necessary to briefly explain the architecture of OpenProdoc.
- The heart of OpenProdoc is its core. The core manages all the logic and functionality of the product.
- The core include "connectors" for managing all its functionality. There are different connectors for each functionality (Metadata management, Document's repositories, Authentication) and it's possible to develop and include new connectors of each class.
- The core itself is very small (<1M) so can be embedded in any application. The Swing and Web clients of OpenProdoc embed the core, and any application (J2EE or not) can embed also the core.

Description of OpenProdoc (II)



Description of OpenProdoc (III)



Test Criteria (I)

- The test is based in a simple program developed for the test, not in the OpenProdoc clients. There are several reasons for this election:
  - The Swing client is not valid due its single user/thread nature.
  - A test with the Web client deployed in any J2EE application server mixes the J2EE server performance, the simulated "browser" performance and the OpenProdoc Engine performance.
  - Additionally, the use of ajax distorts this kind of test.
  - The OpenProdoc core can be embedded in any application.
- So the solution selected is to test the OpenProdoc core with a multithread development.
- This shows the OPD raw performance and makes very easy to test different combinations and also to reproduce the tests in any environ and installation.

Test Criteria (II)

- Additionally, many ECM products include some kind of "Bulk Loader".
- When a project requires to load thousands or hundreds of thousands of documents every day, it's unusual to use the standard client. A Scanning capture system, a massive loader, some kind of development or EAI is used.
- In OpenProdoc, the API used in the Load Test is the standard API, not a program using some "backdoor access". So this is the performance that any development or integration can obtain.
- Therefore, the tests measure the speed and the influence of different factors in the performance of the OpenProdoc core.

Test Criteria (III)

- What are the factors studied?
  1. Running the core and the server in the same machine or different machines.
  2. Different number of threads.
  3. Document types of different complexity.
  4. Different size of documents.
  5. Different database servers.
  6. Different kind of repositories
- In every test, it's measured the speed of four common operations over documents:
  - Insert
  - Download
  - Delete
  - Purge
- Also it's monitored the JVM running the OpenProdoc core to show the load over the computer.

Test Criteria (IV)

- The test were intended to be repeatable and "real", no a "laboratory test", so:
  - The databases used are two Open Source Databases:
    - HSQLDB (used in the portable version of OpenProdoc)
    - MySQL (Widely used in many projects)
  - The databases are used "out-of-the-box", without any fine-tuning.
  - The computers are desktop and portable PC with very standard specifications, no dedicated servers.
  - The Lan is connected using a home ADSL router.

Infrastructure (I)

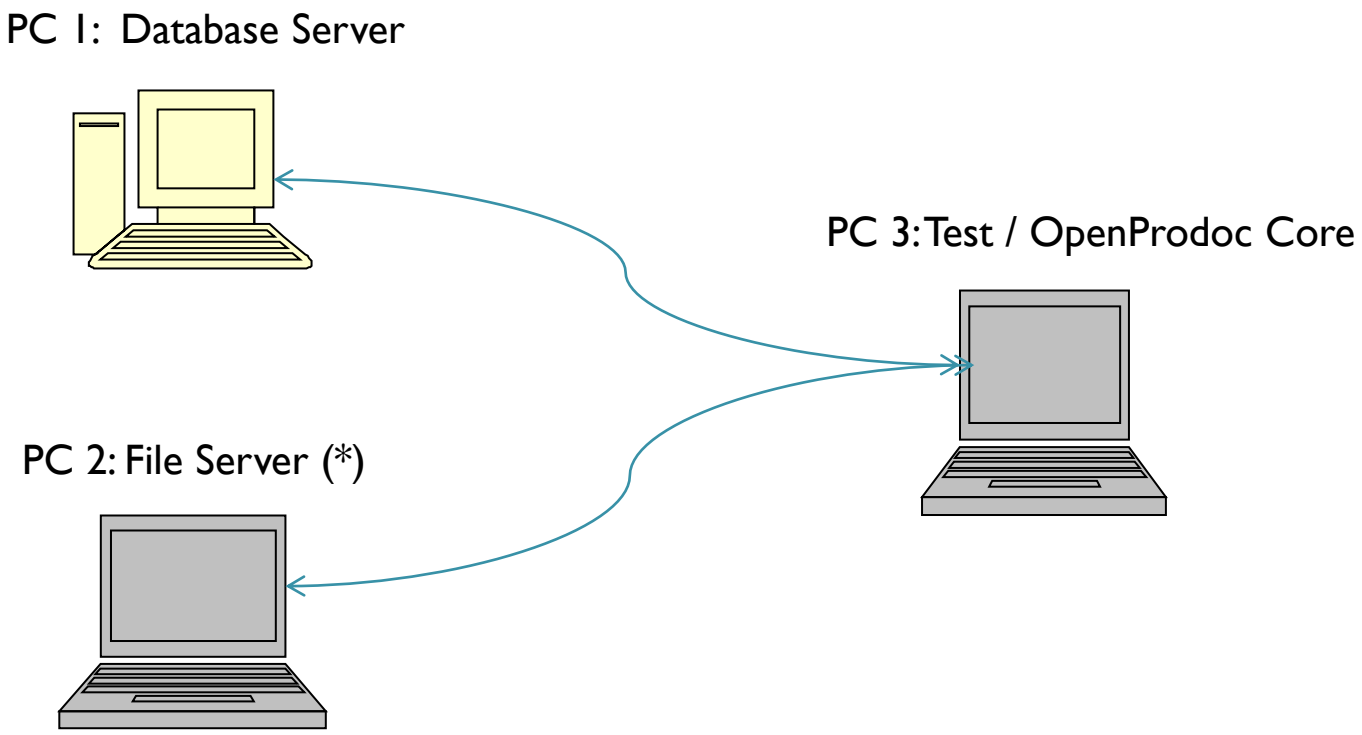
- Test series with 1 PC:
- PC 1:
  - OS: Linux Mint 14. kernel 3.5 32 bits
  - CPU: AMD Phenom II X4 955. 4 nucleus 1.6 GHz
  - RAM: 4 G.
  - Disc: 500G SATA
  - Java 1.6.0.22
- Functions:
  - Database server (HSQLDB 2.2.8, MySQL 5.5.29)
  - Repository server (File System)
  - Test-OpenProdoc run (OpenProdoc 0.8)

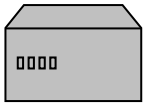
Infrastructure (II)

- Test series with 3 PC:
- PC 1:
  - OS: Linux Mint 14. kernel 3.5 32 bits
  - CPU: AMD Phenom II X4 955. 4 nucleus 1.6 GHz
  - RAM: 4 G, Disc: 500G SATA
  - Java: 1.6.0.22
  - Function: Database server (HSQLDB 2.2.8)
- PC 2:
  - OS: Windows 7 Profesional 32bits
  - CPU: Intel Centrino Core Duo T7500 2 nucleus 2.2GHz
  - RAM: 3 G, Disc: 200G
  - Function: Repository server (FileSystem)
- PC 3:
  - OS: Windows 7 Enterprise 32bits
  - CPU: Intel Core i5. 4 nucleus 2.4GHz
  - RAM: 4 G, Disc: 250G
  - Java: 1.6.0.22
  - Function: Test-OpenProdoc run (OpenProdoc 0.8)

Infrastructure (III)

- Test series with 3 PC:





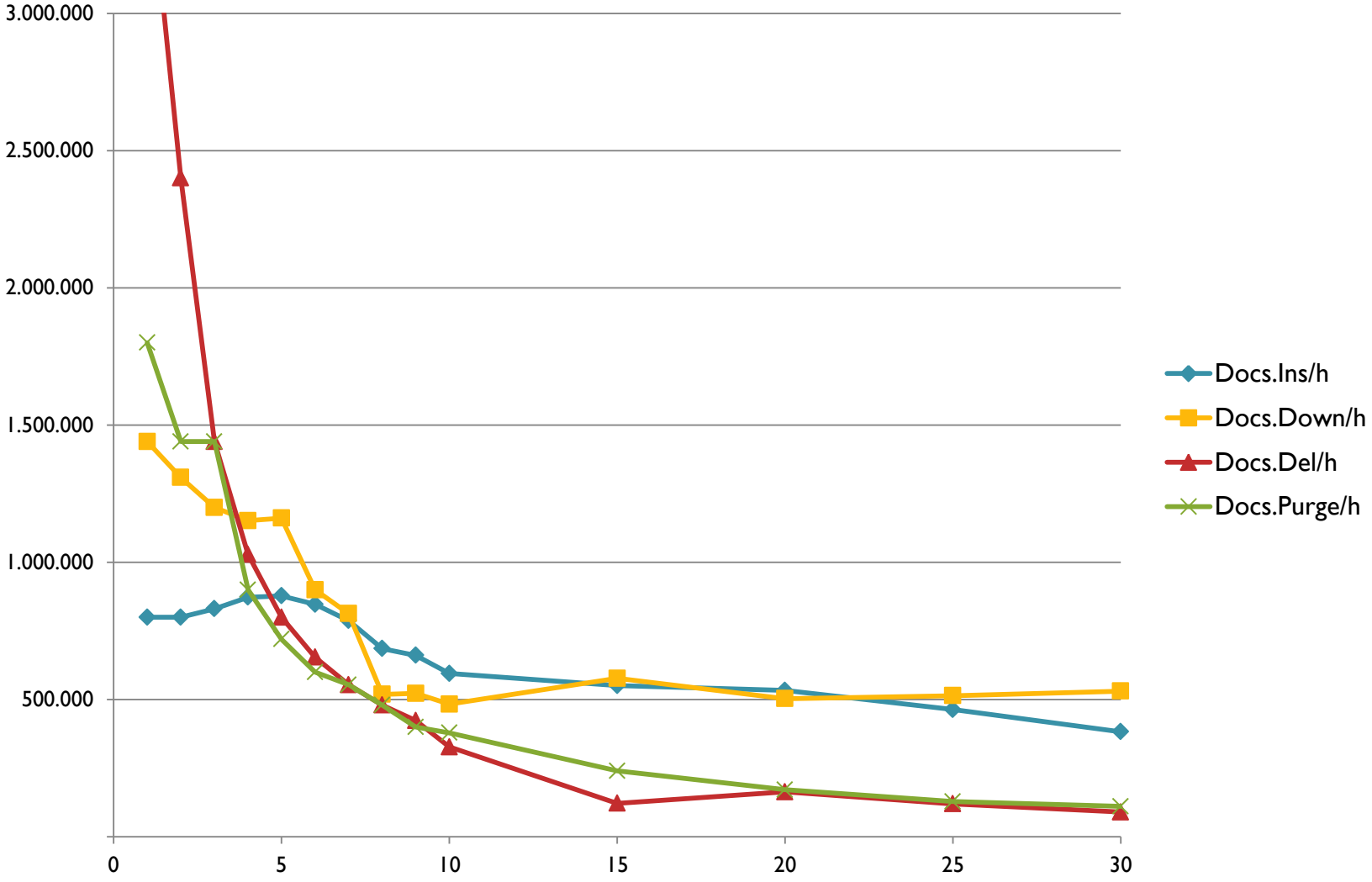
* It was intended to use a (home) NAS disk as File server, but the measures showed a very low speed. (x4 slower than PC 2). Some Internet investigations showed that the family of disk has some speed problems, so the test results were discarded.

Realization of the Test

- The program developed for the test, executes the instructions:
  - Reads the configuration file, that details the document to be inserted, the number of threads, the number of documents to be used and the document type.
  - Creates local copies of the source documents, so that they are inserted from different files and not always from the same file in cache.
  - Creates a folder in OpenProdoc for storing the folders and documents of the tests.
  - For every set of tests:
    - Starts the specified number of threads for insertion.
    - Measures the time **from the start of the first thread to the end of the last thread**.
    - Repeats the process for download threads, delete threads and purge threads.
    - Saves the measured data
    - Deletes the downloaded documents
  - Deletes the local copies of the document used for insertion.

Results and Graphs. Local Tests (I)

- Test managing from 1 to 30 threads, 2.000 to 60.000 docs of 100k

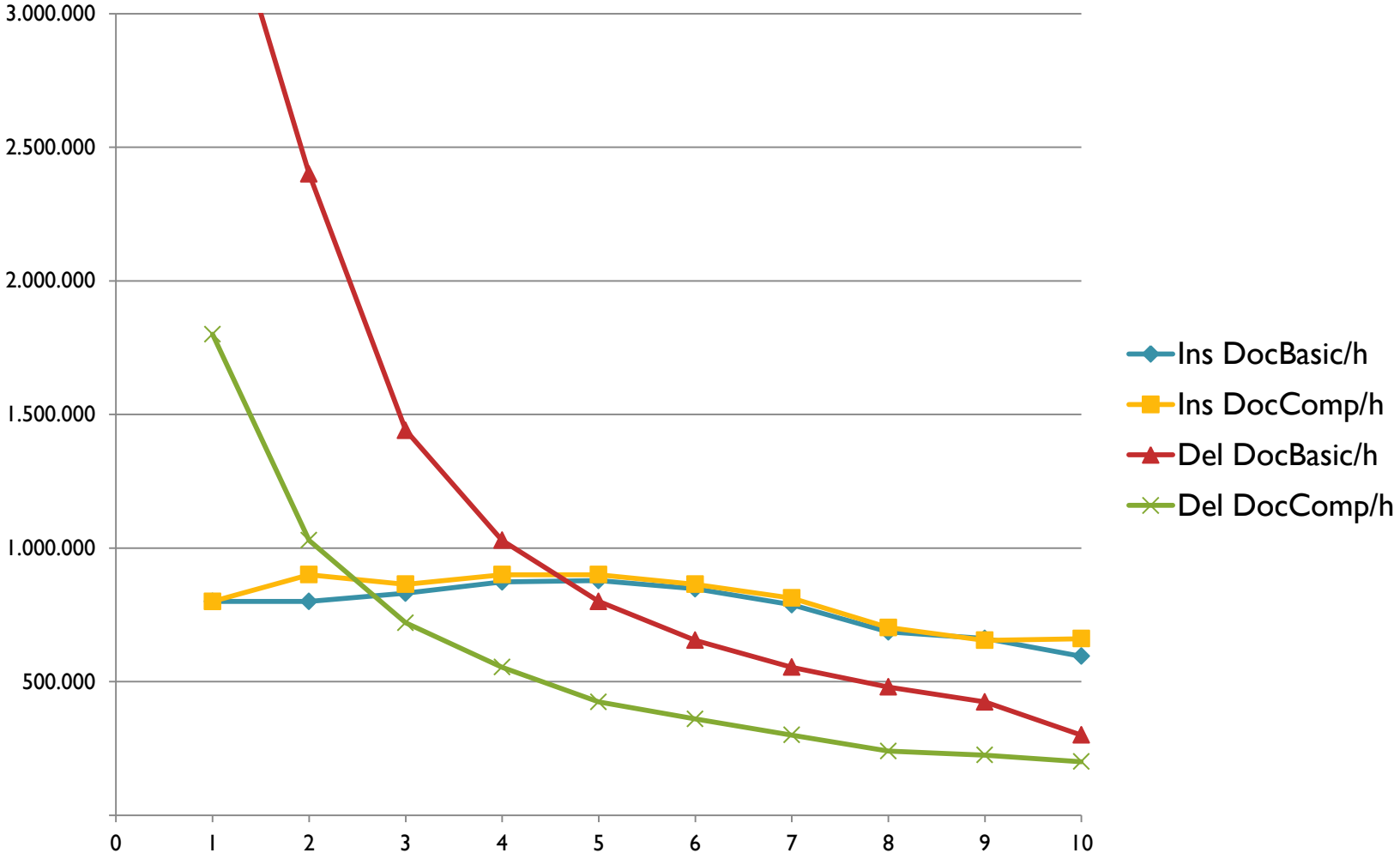


Results and Graphs. Local Tests (II)

- The graph shows the times measured running the OpenProdoc core, the database and the repository in one computer.
- The maximum number of documents inserted (878.000 docs/h) is obtained using 5 threads.
- The Download, Purge and Delete functions decrease with the number of threads, but even with 30 threads, the purge and delete rate is around 100.000 docs/h.
- This configuration has the advantage of avoiding communications, but the disk and CPU have a higher load.

Results and Graphs. Doc. Type (I)

- Test managing from 1 to 10 threads, 2.000 to 20.000 docs of 100k in two types

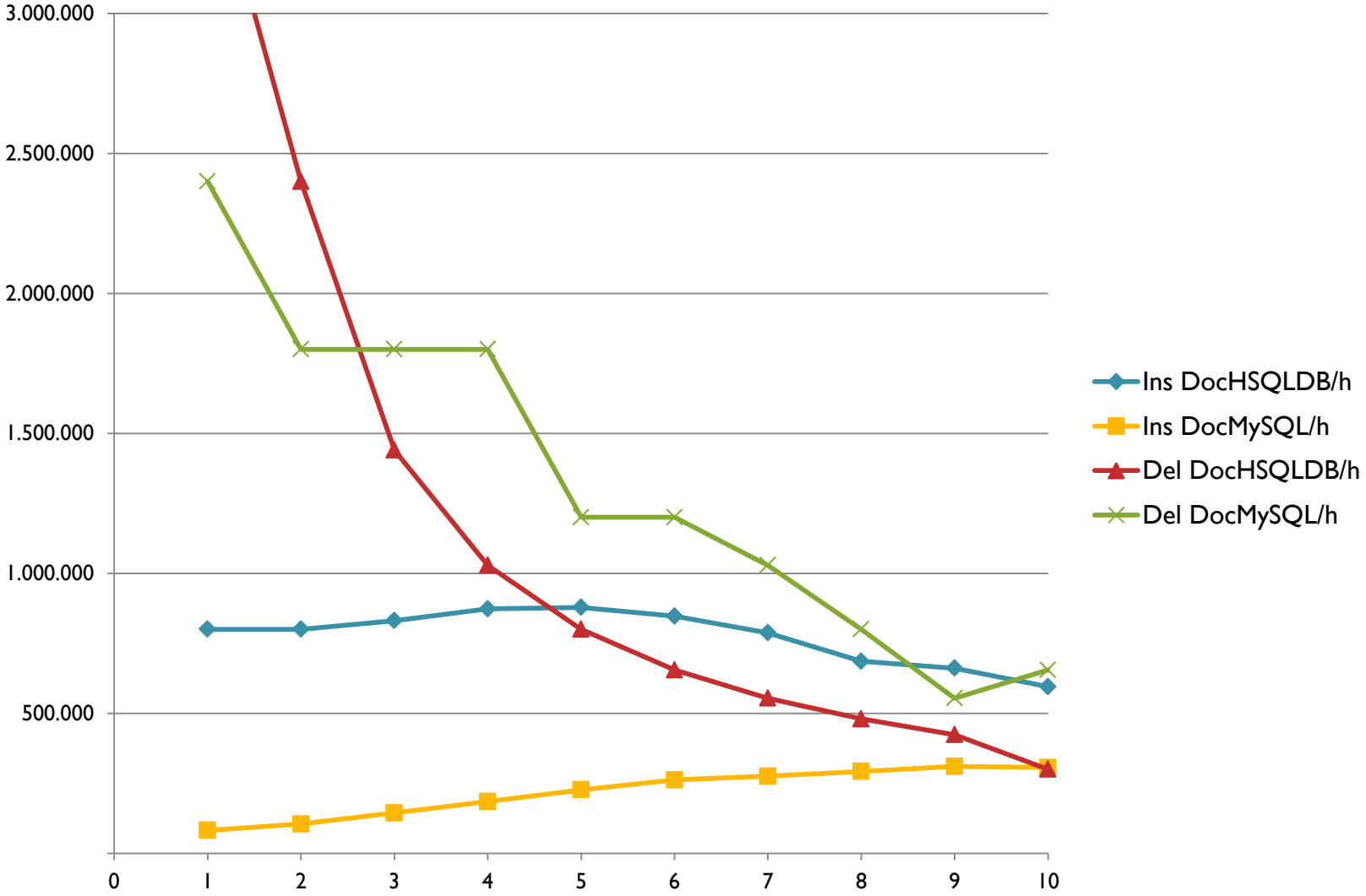


Results and Graphs. Doc. Type (II)

- The graph shows the times measured running the OpenProdoc core, the database and the repository in one computer with two document types, the basic type and a compound type, subtype of the first one.
- The document types in OpenProdoc are structured as related tables, so when the hierarchy level grows, the operations affect to more tables and are slower.
- The insertion speed is very similar because the time is used mainly in the upload and storing of the file in the repository.
- Operations as delete or queries are slower for complex types because internally OpenProdoc needs to access more tables.

Results and Graphs. HSQLDB vs MySQL (I)

- Test managing from 1 to 10 threads, 2.000 to 20.000 docs of 100k in two DD.BB.

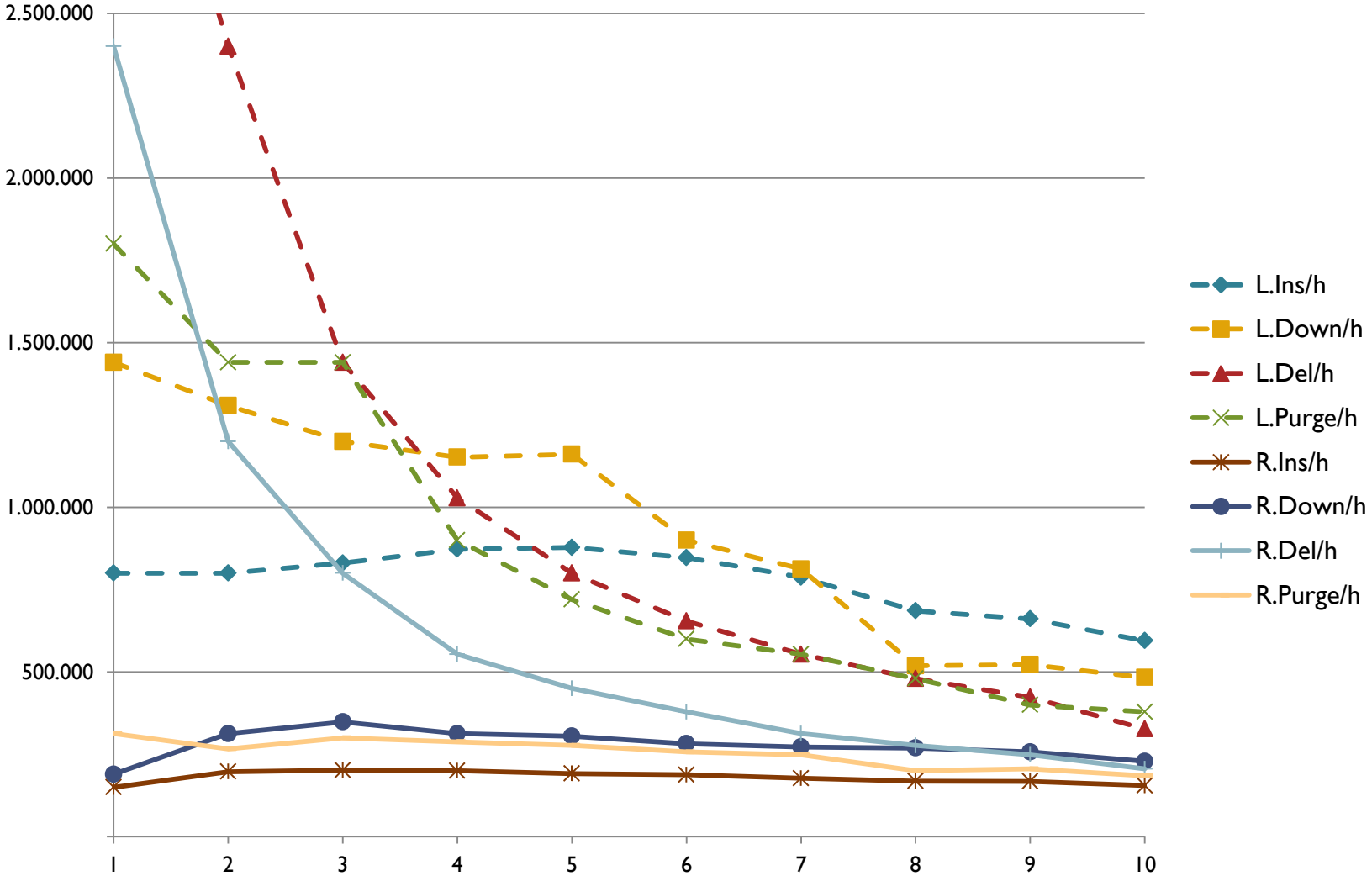


Results and Graphs. HSQLDB vs MySQL (I)

- The graph shows the times measured running the OpenProdoc core, the database and the repository in one computer with different database servers.
- With the default configuration, the performance obtained with HSQLDB is better than with MySQL, not only in operation with documents management but in operations with metadata management (the Delete only moves documents to the trash bin, without actually deleting the file, that is eliminated with the Purge).
- MySQL maintains the performance when increasing the number of threads, however HSQLDB decreases the performance, although is always better than MySQL.
- Of course, with fine tuning of the databases, the values probably can change, but it's out of the scope of this tests.

Results and Graphs. Local vs Remote (I)

- Test managing from 1 to 10 threads, 2.000 to 20.000 docs of 100k in two architectures

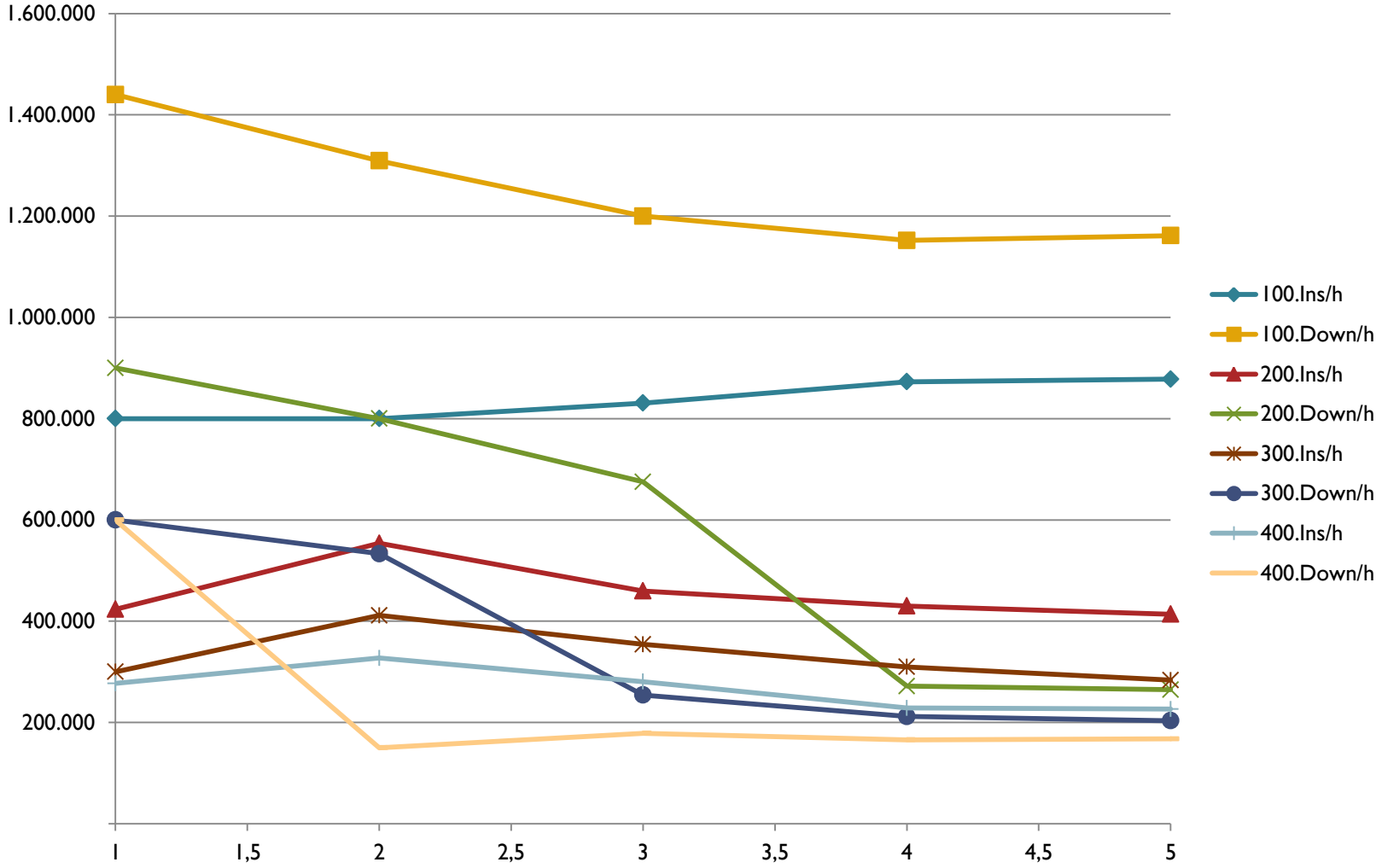


Results and Graphs. Local vs Remote (I)

- The graph shows the times measured running the OpenProdoc core, the database and the repository in one computer (local) or in three different computers.
- In general, the speed when all the elements are in the same computer is around 4 times the speed with the elements distributed in different computers.
- The only exception is the delete function due that is related only to database and not to repository/filesystem.
- This shows that a fast filesystem is the key to improve the performance with this architecture.
- Anyway, the insert speed is 200.000 docs/h, the download speed is 348.000 docs/h and the purge speed is 300.000 docs/h, enough throughput for most of the installations.

Results and Graphs. Document Size (I)

- Test managing from 1 to 5 threads, 2.000 to 10.000 docs of 100k, 200k, 300k and 400k

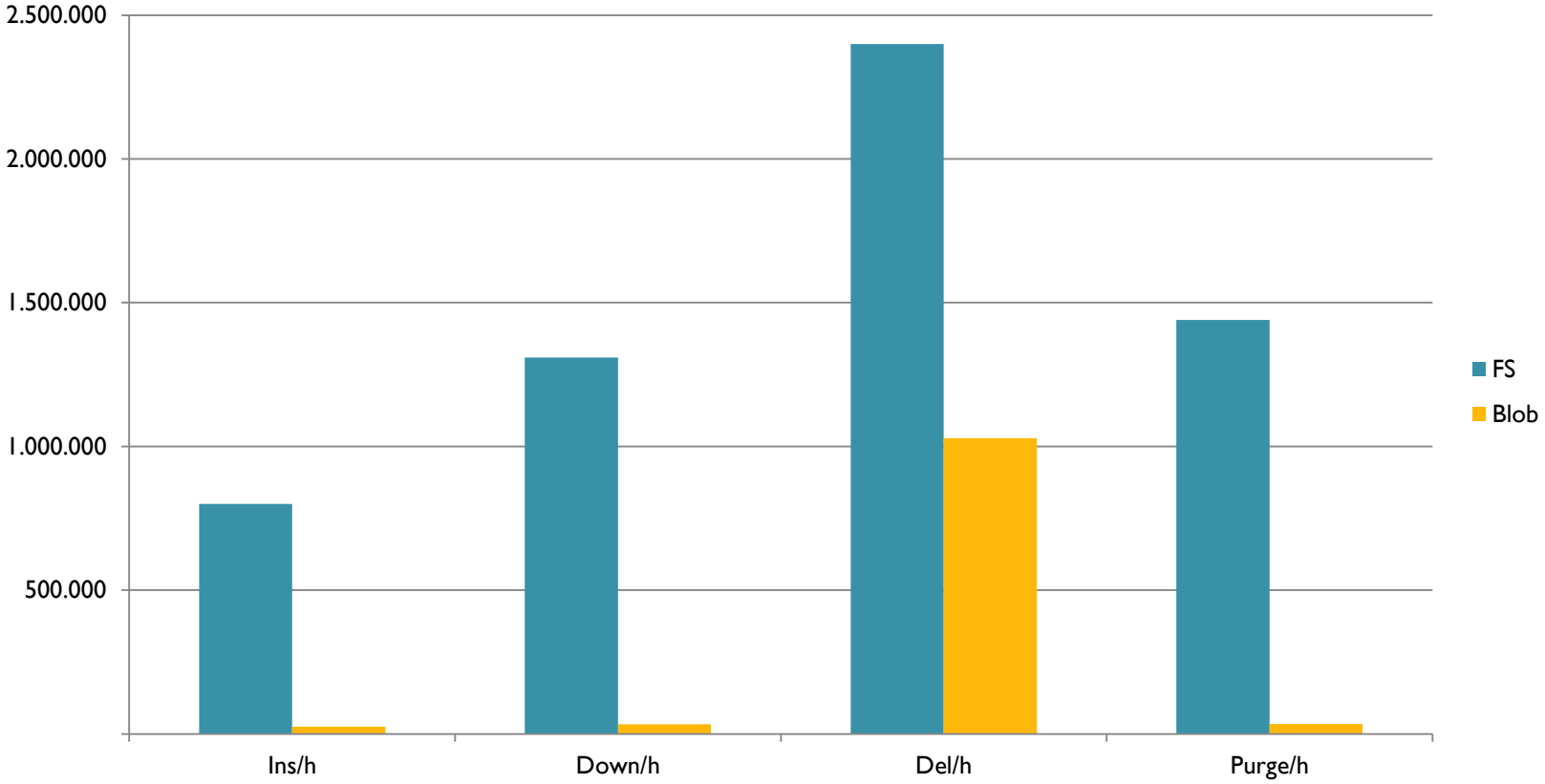


Results and Graphs. Document Size (II)

- The graph shows the times measured running the OpenProdoc core, the database and the repository with document of different sizes.
- The delete operations affect only to the metadata and therefore the speed is similar, with normal statistics variations.
- The purge operations (with file system repository) also have little relation with the file size.
- The insert and download operations show a decrease of speed with the file size, but without an apparent coefficient. Probably the execution of the sets of tests was affected by the garbage collector in the database server (Java alone) creating the "steps" in the data.
- Repeating the group of test, appeared also some "steps" but in different places.

Results and Graphs. Repositories (I)

- Test managing 2 threads, 4.000 docs of 100k in two repositories



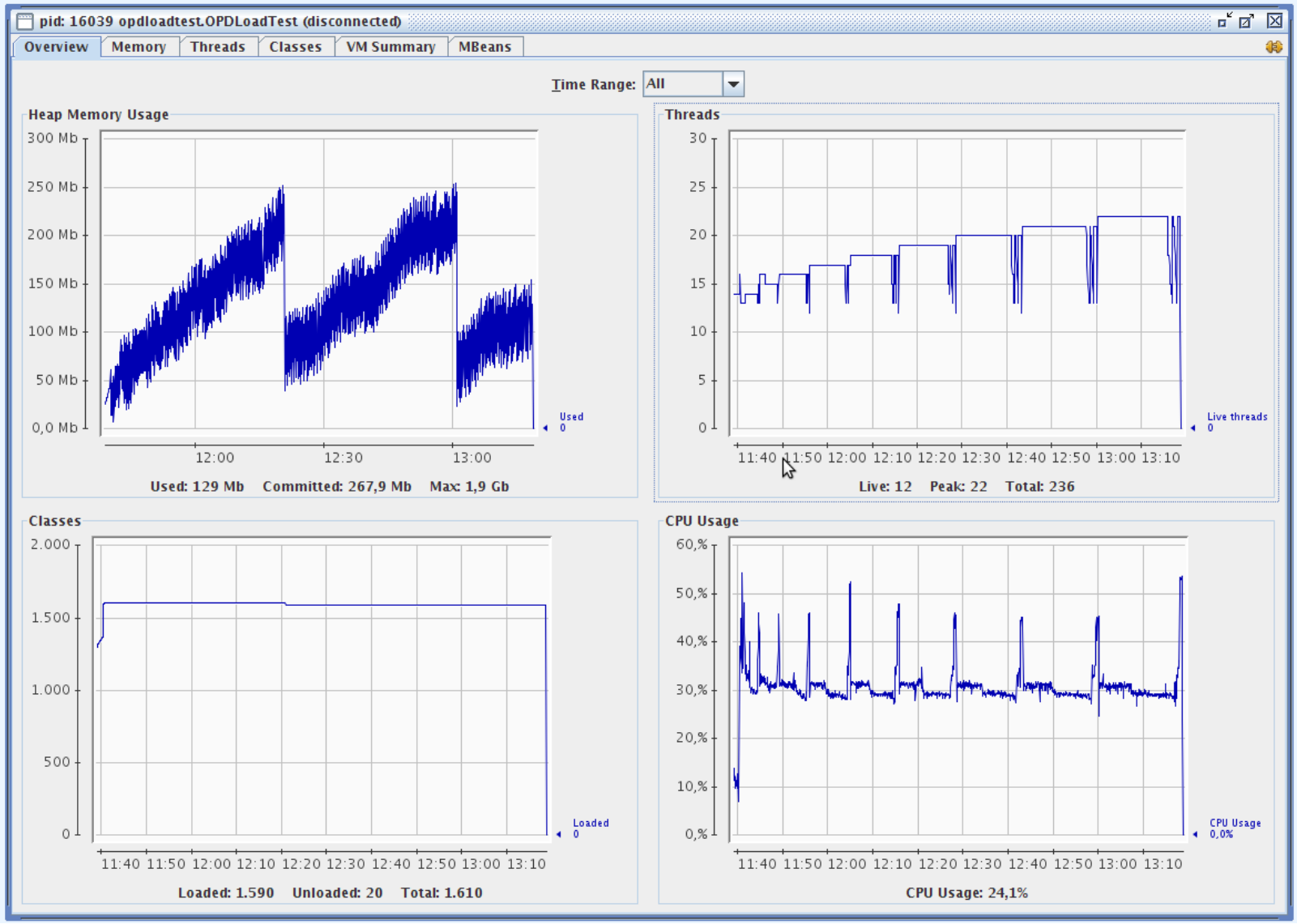
	Ins/h	Down/h	Del/h	Purge/h
FS	800.000	1.309.091	2.400.000	1.440.000
Blob	24.870	33.488	1.028.571	34.286

Results and Graphs. Repositories (II)

- The graph shows the times measured running the OpenProdoc core, the database and two repositories, a file system and a database Blob.
- The speed with Blob is very low compared with the file system speed.
- Even the delete speed, that doesn't affect the actual storage is slower, probably by the general overload of the database.
- The speed of blob storage can change a lot between different databases due to its different implementation.
- Anyway, 25.000 docs/hour is enough for many systems. Some scenarios with small documents (icons, SMS, Twitter messages, IM messages, email, etc.) can be a use case.

Results and Graphs. Java Jvm (I)

- Jvm monitor. Test of 10 series managing from 4.000 to 40.000 docs of 200k



Results and Graphs. Java Jvm (II)

- The JVM contains the test program, the OpenProdoc Core and the HSQLDB client, so the numbers contain added values (in Class number, threads, heap, etc.) from both elements.
- This would show what the overhead of embedding OpenProdoc with the HSQLDB client would be for an application.
- Every set of tests can be observed by the peaks in CPU use (due to creating threads) and in the total number of threads.
- The heap is always under 256 M and has a sudden drop when de garbage collector starts.
- In this set of tests, 220.000 documents of 200k were inserted, downloaded and deleted.

Analysis and Conclusions (I)

- The objectives of the test where:
  - To check the behaviour of OpenProdoc under stress conditions, detecting potential problems.
  - To measure the speed to manage documents
  - To detect the influence of different factors in the performance of OpenProdoc.
- Regarding the first point, the stability, resource utilization and overall functioning has been correct. The tests discovered a potential error in date metadata for high concurrence scenarios that has been corrected after the first set of test.
- The concurrency obtained with the test program is similar to a standard use of hundreds or thousands of users.

Analysis and Conclusions (II)

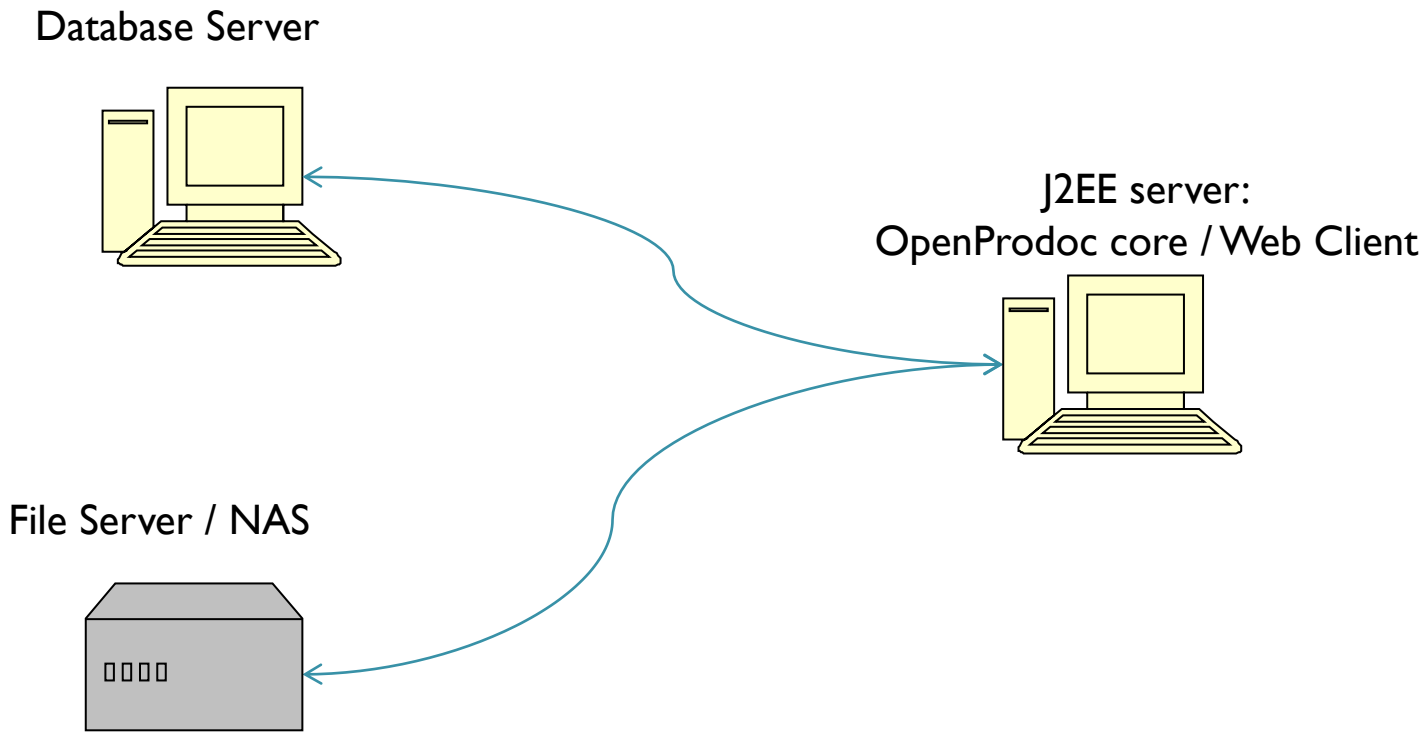
- Regarding to speed, the obtained processing speed is high, especially considering that OpenProdoc is a document management system fully functional, with permissions for each document (not only document type or folder), user permissions, integrity and transaction management.
- The influence of the different factors is more or less the expected, knowing the internal model and design of OpenProdoc, but after the tests there is a model that allows a more realistic estimation.
- Perhaps the differences in performance between databases is striking, but probably, modifying the default configuration the differences can be reduced.

Using OpenProdoc (I)

- The recommended infrastructure for using OpenProdoc will be determined by different criteria.
- The tests show that even with a small infrastructure is possible to obtain a high throughput, so the performance isn't is the main factor.
- Using it embedded in an application or with its own Web Client there are no big requirements, so the decision will be determined mainly by company standards (O.S., Database, etc.), security criteria and architecture requirements (High Availability, scale up, scale out, etc.).
- A minimum configuration can be a computer with O.S. Linux 64 bits 8G Ram with the functionality of database server (HSQLDB o MySQL), repository (file system) and application server (Tomcat o Glassfish).

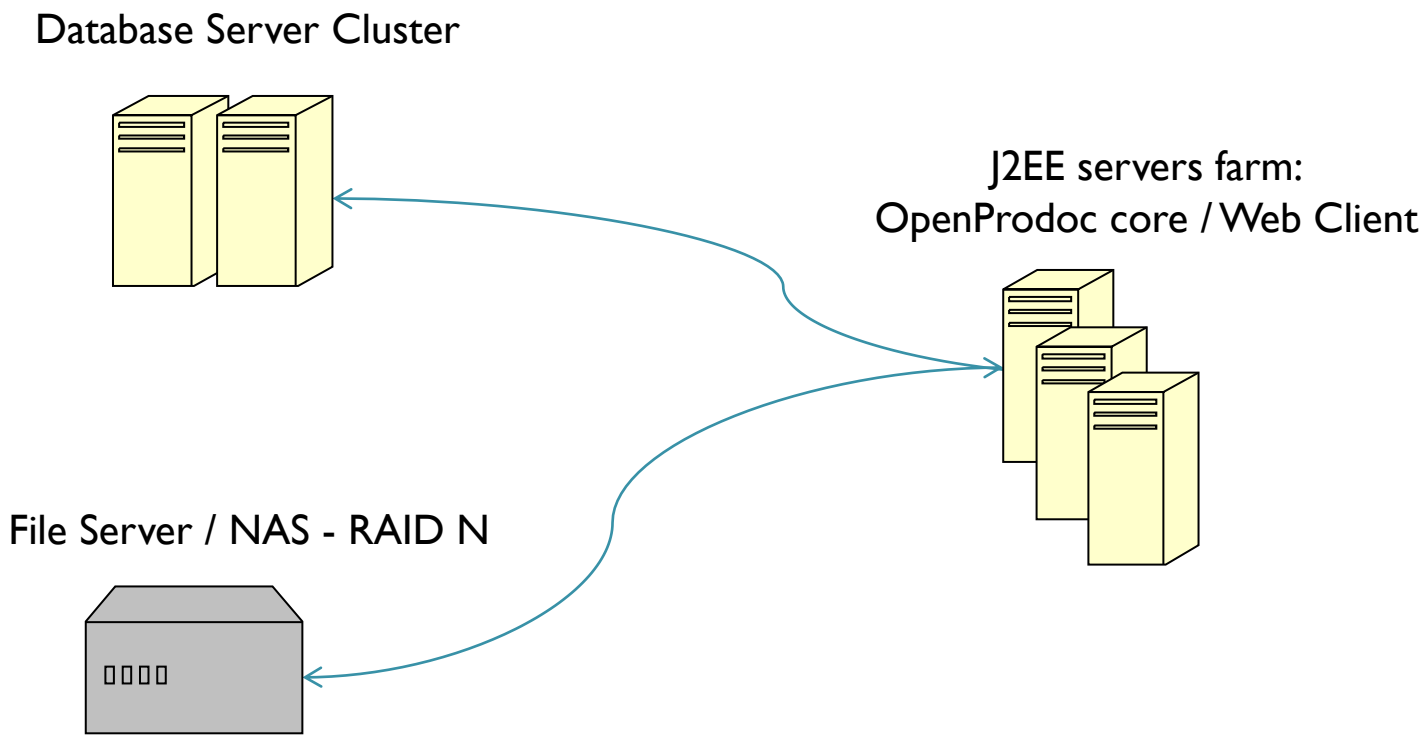
Using OpenProdoc (II)

- A more standard configuration, sharing all the servers with other applications and services, would be:



Using OpenProdoc (III)

- Finally, a Enterprise solution can be similar to:



More Information

- http://code.google.com/p/openprodoc/

- Joaquin Hierro
- openprodoc@gmail.com