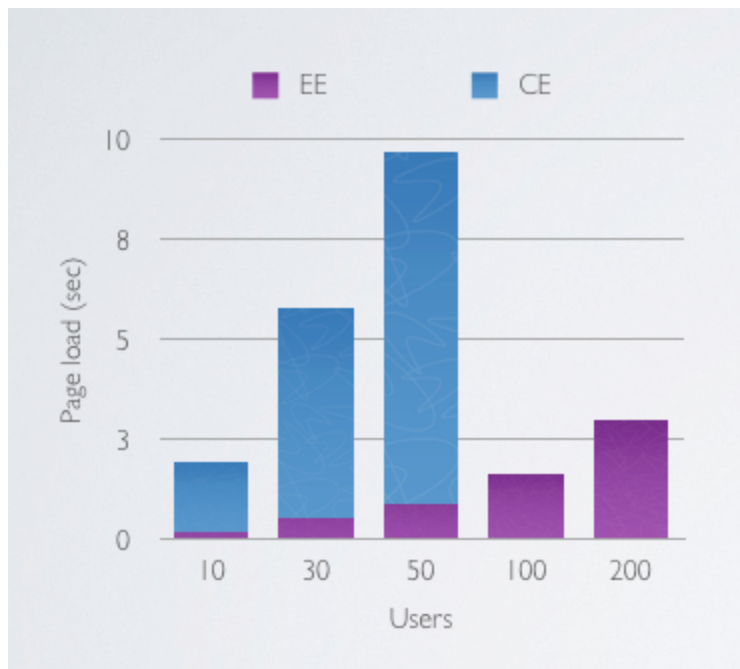


## Magento Performance And optimization

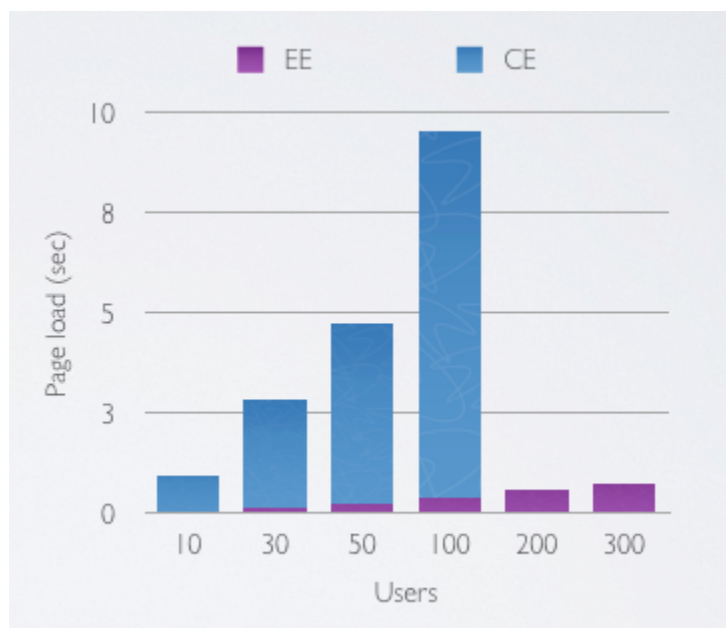
- **APC** – most of the RAM consumption goes on loading and interpreting the PHP files. Bytecode cache accelerators improve tremendously both execution time and memory consumption by caching the compiled code. In our tests with APC the RAM consumption per thread was down to 2-5MB, and speed improvement up to 6 times.
- **HTML block cache** – Magento has option to specify for each block Cache key and expiration time. We plan gradually implement caching on all suitable blocks by finding the optimal and when needed configurable values for both parameters.
- **Custom Resource Models** – Magento supports an ability to create a custom backend logic to communicate with low level resources (database). That means creation of database structure optimized for your operations is possible. It is even possible to create compiled executable to handle your business logic, as long as they support original resource model API.
- **SQL cluster** – Magento was built from group up to support separate read and write connections. Meaning without much hassle you can set up database cluster with one master for writes and multiple slaves for read connections.
- **HTTP load balancing** – This is a standard way to spread load over multiple web servers, no major Magento customization is needed.  
Now, some math. Assume we are using APC and with Magento cache is enabled, on our **development server** (1x2GHz Xeon, 2GB RAM) gave around 0.05-0.2sec, 2-7MB per request, for safety we'll take 8MB per request.  
**During normal load** – 1K/minute - 17 views per second – even if a thread takes 1 sec to complete there will be  $8M \cdot 17 = 136MB$ .  
**During peak load** – 20K/minute – 334 views per second –  $8MB \cdot 334 = 2.672GB$  – split between 2-4 load balanced servers with 2GB each should be ok.

### Magento Performance Graph (CE vs EE)

Default config with Apache



Config with APC cache Handler (Explained Above)



### Distributed caching Using Memcached

Memcached is a distributed memory caching system that stores data retrieved from the

database (or other data source) in memory, allowing repeat reads to retrieve the data from memory rather than needing to query the database again, greatly increasing the speed of web applications.

### **MySql Splitting :**

Magento has got a cool feature for coping with a *database replicating facility*. Replication can be used in many cases to build effective and scalable, highly available solutions. There are a number of different methods for setting up replication, and the exact method to use depends on how you are setting up replication, and whether you already have data within your master database.

### **Apache Web Server Tuning :**

Some of the apache module is important in magento optimization

- 1 - Apache KeepAlive
- 2 - Apache mod\_cache
- 3 - Apache mod\_expires

### **GZip Compression**

Web pages can be compressed using GZip between the server and the client, reducing the amount of data that needs to be transferred. Although the act of compressing and decompressing adds a performance overhead there is usually a net gain in reducing the amount of traffic especially for large pages.

### **Varnish Cache:**

Varnish is an HTTP accelerator designed for content-heavy dynamic web sites. In contrast to other HTTP accelerators, such as Squid, which began life as a client-side cache, or Apache, which is primarily an origin server, Varnish was designed from the ground up as an HTTP accelerator.

### **CDN**

The distance between webserver and browser also determines how long the visitor must wait before the webpage is loaded, and this distance can be shortened by using a CDN: The webserver that is the closest to the visitor is serving the content. Things like hops and QoS are

involved here

**There are tons of other options for tuning magento (MYSQL,Apache,PHP) which are dependent upon the requirement**