Article
Murray Oldfield · Oct 1, 2016 10m read

Data Platforms and Performance - Part 7 ECP for performance, scalability and availability

One of the great availability and scaling features of Caché is Enterprise Cache Protocol (ECP). With consideration during application development distributed processing using ECP allows a scale out architecture for Caché applications. Application processing can scale to very high rates from a single application server to the processing power of up to 255 application servers with no application changes.

ECP was used widely for many years in TrakCare deployments I was involved in. A decade ago a 'big' x86 server from one of the major vendors might only have a total of eight cores. For larger deployments ECP was a way to scale out processing on commodity servers rather than a single large and expensive big iron enterprise server. Even the high core count enterprise servers had limits so ECP was also used to scale deployments on them as well.

Today most new TrakCare deployments or upgrades to current hardware *do not require ECP* for scaling. Current two-socket x86 production servers can have dozens of cores and huge memory. We see that with recent Caché versions TrakCare -- and many other Caché applications -- have predictable linear scaling with the ability to support incremental increases in users and transactions as CPU core counts and memory increase in a single server. In the field I see most new deployments are virtualised, even then VMs can scale as needed up to the size of the host server. If resource requirements are more than a single physical host can provide then ECP is used to scale out.

- ***Tip:*** *For simplified management and deployment scale within a single server before deploying ECP.*

In this post I will show an example architecture and the basics of how ECP works then review performance considerations with a focus on storage.

Specific information on configuring ECP and application development is available in the online Caché Distributed Data Management Guide and there is an ECP learning track here on the community.

One of the other key features of ECP is increasing application availability, for details see the ECP section in the Caché high availability guide.

A list of other posts in this series is here

ECP Architecture Basics

The architecture and operation of ECP is conceptually simple, ECP provides a way to efficiently share data, locks, and executable code among multiple server systems. Viewed from the application server data and code are stored remotely on a *Data server*, but are cached in memory locally on the *Application servers* to provide efficient access to active data with minimal network traffic.

The Data server manages database reads and writes to persistent storage on disk while multiple Application servers are the workhorses of the solution performing most of the application processing.

Multi-tier architecture

ECP is a multi-tier architecture. There are different ways to describe processing tiers and the roles they perform, the following is what I find useful when describing web browser based Caché applications and is the model and terminology for my posts. I appreciate that there may be different ways to break down tiers, but for now lets use my way :)

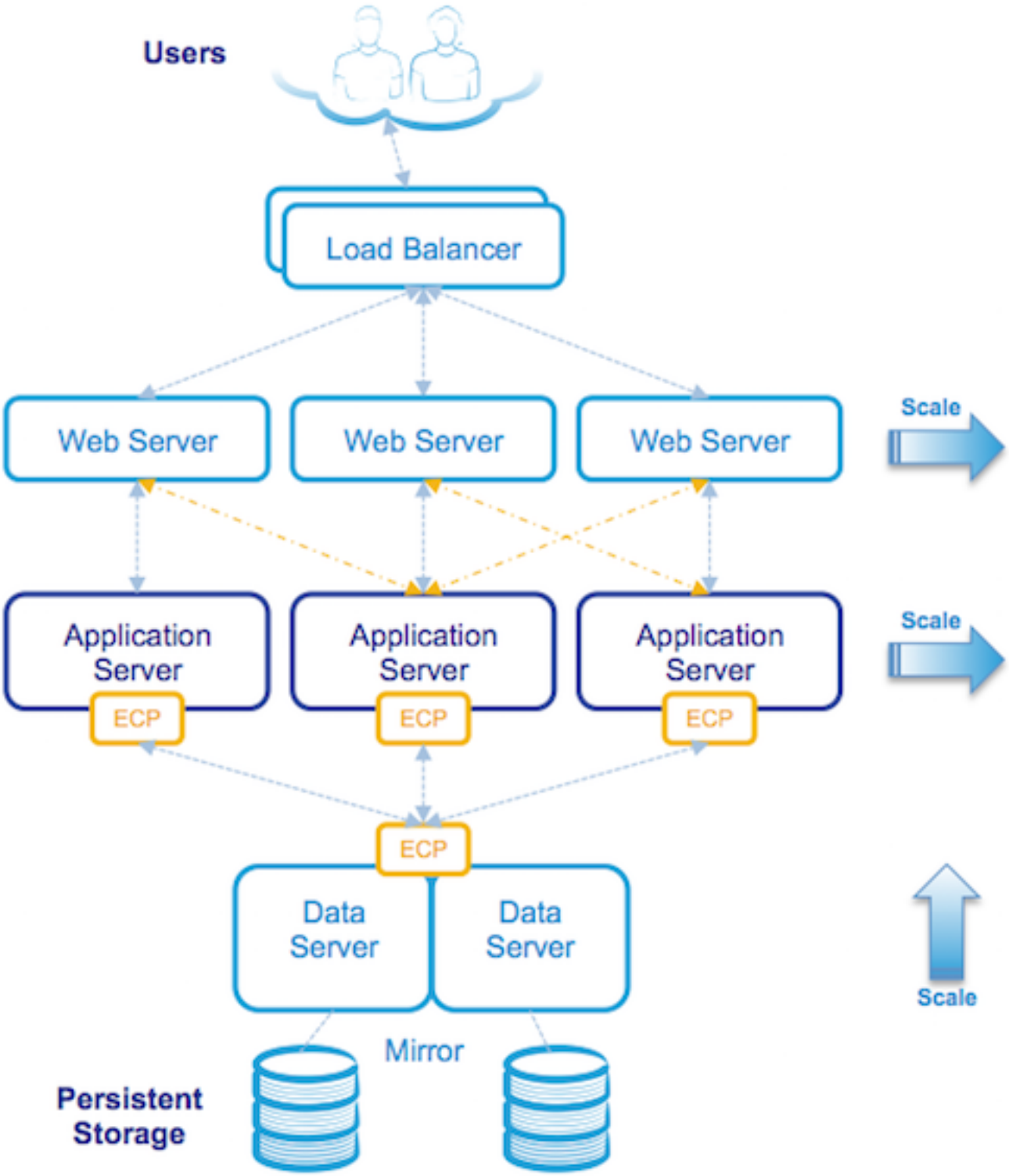
A browser based application, for example using Caché Server Pages (CSP) uses a multi-tier architecture where presentation, application processing, and data management functions are logically separated. **Logical** 'servers' with different roles populate the tiers. Logical servers do not have to be kept on separate physical host or virtual servers, for cost effectiveness and manageability some or even all logical servers may be located on a single host or operating system instance. As deployments scale up servers may be split out to multiple physical or virtual hosts with ECP so spreading the processing workload as needed without change to the application.

Host systems may be physical or virtualised depending on your capacity and availability requirements. The following tiers and logical servers make up a deployment:

- *Presentation Tier:* Includes the Web Server which acts as gateway between the browser-based clients and the application tier.
- *Application Tier:* This is where the ECP Application server sits. As noted above this is a logical model where the application server does not have to be separate from the Data server, and are typically not required to be for all but the largest sites. This tier may also include other servers for specialised processing such as report servers.
- *Data Tier:* This is where the Data server is located. The data server performs transaction processing and is the repository for application code and data stored in the Caché database. The Data Server is responsible for reading and writing to persistent disk storage.

Logical Architecture

The following diagram is a logical view of a browser based application when deployed as a three-tier architecture:



Although at first glance the architecture may look complicated it is still made up of the same components as a Caché system installed on a single server, but with the logical components installed on multiple physical or virtual servers. All communication between servers is via TCP/IP.

ECP Operation in the logical view

Starting from the top the diagram above shows users connecting securely to multiple load balanced web servers. The web servers pass CSP web page requests between the clients and the application tier (the Application servers) which perform any processing, allowing content to be created dynamically, and returns the completed page back to the client via the web server.

In this three-tier model application processing has been spread over multiple Application servers using ECP. The application simply treats the data (your application database) as if it was local to the Application server.

When an Application server makes a request for data it will attempt to satisfy the request from its local cache, if it cannot, ECP will request the necessary data from the Data server which may be able to satisfy the request from its own cache or if not will fetch the data from disk. The reply from the Data server to the Application server includes the database block(s) where that data was stored. These blocks are used and now cached on the Application server. ECP automatically takes care of managing cache consistency across the network and propagating changes back to the Data server. Clients enjoy fast responses because they frequently use locally cached data.

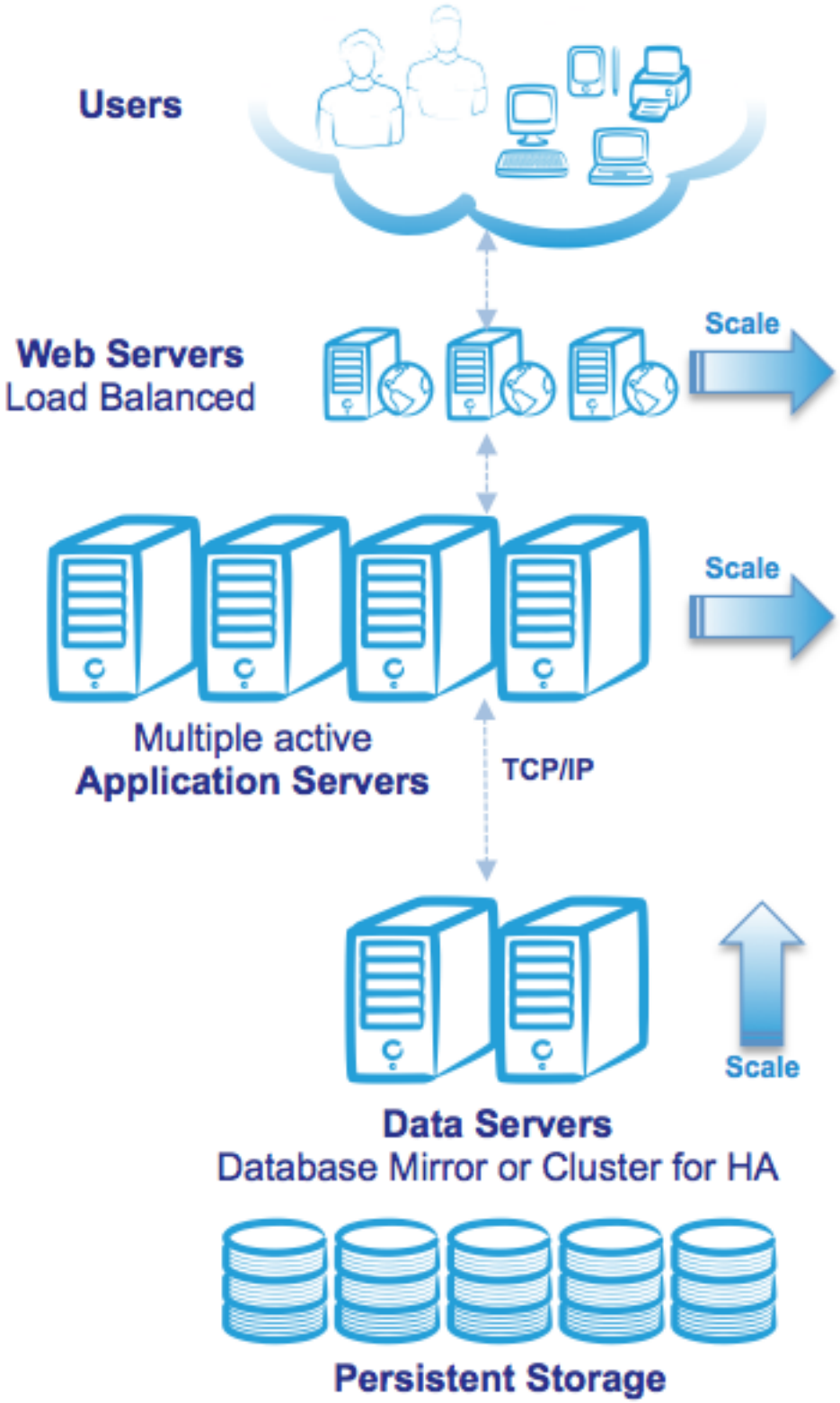
By default web servers communicate with a preferred Application server ensuring that the same Application server services subsequent requests for related data as the data is likely to already be in local cache.

- ***Tip:*** *As detailed in the Caché documentation avoid connecting users to application servers in a round-robin or load-balancing scheme wich impacts the benefit of caching on the application server. Ideally the same users or groups of users stay connected to the same application server.*

The solution is scaled without user downtime at the Presentation Tier by adding web servers and at the Application Tier by adding additional Application servers. The Data tier is scaled by increasing CPU and memory on the Data servers.

Physical Architecture

The following diagram shows an example of physical hosts used in the same three-tier deployment as the three-tier logical architecture example:



Note that physical or virtualised hosts are deployed at each tier using an n+1 or n+2 model for 100% capacity in event of a host failure or scheduled maintenance. Because users are spread across multiple web and application servers, the failure of a single server affects a smaller population with users automatically reconnecting to one of the remaining servers.

The Data management tier is made highly available, for example located on a failover cluster (e.g. virtualization HA, InterSystems Database Mirroring, or traditional failover clustering) connected to one or more storage arrays. In the event of hardware or service failure clustering will restart the services on one of the surviving nodes in the cluster. As an added benefit, ECP has built-in resiliency and maintains transactional integrity in the event of a

database node cluster failover, application users will observe a pause in processing until failover and automatic recovery completes and users then seamlessly resume without disconnection.

The same architecture can also be mapped to virtualised servers, for example VMware vSphere can be used to virtualise Application servers.

ECP Capacity Planning

As noted above the Data server manages database reads and writes to persistent disk while multiple Application servers are the workhorses of the solution performing most of the application processing. This is a key concept when considering system resource capacity planning, in summary:

- The ***Data server*** (sometimes called the Database server) typically performs very little application processing so has *low CPU requirements*, but this server performs the majority of storage IO, so can have *very high storage IOPS* i.e. database reads and writes as well as journal writes (more on journal IO later).
- The ***Application server*** performs most application processing so has *high CPU requirements*, but does very little storage IO.

Generally you size ECP server CPU, memory and IO requirements using the same rules as if you were sizing a very large single server solution while taking into account N+1 or N+2 servers for high availability.

Basic CPU and Storage sizing:

Imagine My_Application needs a peak 72 CPU cores for application processing (remember also accounting for headroom) and is expected to require 20,000 writes during write daemon cycle and a sustained peak 10,000 random database reads.

A simple back of the envelope sizing for virtual or physical servers is:

- 4 x 32 CPU Application servers (3 servers + 1 for HA). Low IOPS requirements.
- 2 x 10 CPU Data servers (Mirrored or Clustered for HA). Low latency IOPS requirement is 20K writes, 10K reads, plus WIJ and Journal.

Even though the Data server is doing very little processing it is sized at 8-10 CPUs to account for System and Caché processes. Application servers can be sized based on best price/performance per physical host and/or for availability. There will be some loss in efficiency as you scale out, but generally you can add processing in server blocks and expect a near linear increase in throughput. Limits are more likely to be found in storage IO first.

- ***Tip:*** *As usual for HA consider the effect of host, chassis or rack failures. When virtualising Application and Data servers on VMWare ensure vSphere DRS and affinity rules are applied to spread processing load and ensure availability.*

Journal synchronisation IO requirements

An additional capacity planning consideration for ECP deployments is they require higher IO and impose a very stringent storage response time requiremenst to maintain scalability for journaling on the Data server due to journal synchronisation (a.k.a. a journal sync). Synchronisation requests can trigger writes to last block in the journal to ensure data durability.

Although your milage may vary; at a typical customer site running high transaction rates I often see journal write IOPS on non ECP configurations in the 10's per second. With ECP on a busy system you can see 100's to 1,000's of write IOPS on the journal disk because of the ECP imposed journal sync's.

- ***Tip:*** *If you display mgstat or look at mgstat in pButtons on a busy system you will see Jrnwrts (Journal Writes) which you will be accounting for in your storage IO resource planning. On an ECP Data server there are also Journal Synchronistion writes to the journals disk that are not displayed in mgstat, to see these you will need to look at operating system metrics for your journal disk, for example with iostat.*

What are journal syncs?

Journal syncs are necessary for:

- Ensuring data durability and recoverability in the event of a failure on the data server.
- They also are triggers for ensuring cache coherency between application servers.

In non-ECP configurations modifications to a Caché database are written to journal buffers (128 x 64K buffers) which are written to journal files on disk by the journal daemon as they fill or every two seconds. Caché allocates 64k for an entire buffer, and these are always re-used instead of destroyed and recreated and Caché just keeps track of the ending offset. In most cases (unless there are a massive updates happening at once) the journal writes are very small.

In ECP systems there is also journal synchronisation. A journal sync can be defined as re-writing the relevant portion of the current journal buffer to disk to ensure the journal is always current on disk. So there are many re-writes of a portion of the same journal block (anywhere between 2k and 64k in size) from journal sync requests.

Events on an ECP client that can trigger a journal sync request are updates (SET or KILL), or a LOCK. For example for each SET or KILL the current journal buffer is written (or rewritten) to disk. On very busy systems journal syncs can be bundled or deferred into multiple sync requests in a single sync operation.

Capacity planning for journal syncs

For sustained throughput average write response time for journal sync must be:

- *<=0.5 ms with maximum of <=1 ms.*

For more information see the IO requirements table in this post: Part 6 - Caché storage IO profile.

- ***Tip:*** *When using Caché Database Mirroring with ECP journal syncs are applied on both primary and backup mirror node journal disks. This should not be a concern as a rule of mirror configuration is both nodes will be configured as equals for storage IO.*

You will have to validate specific IO metrics for you own systems, the aim of this section is to share with you that there are very strict response time requirements and understanding where to look for metrics.

- ***Tip:*** *On Red Hat Linux and SuSE XFS filesystem is recommended for Caché systems. However testing has shown that ext4 provides higher IOPS capability for small writes like journal synchs. Consider using ext4 for journal disks.*

Summary

This post is an orientation to ECP and additional metrics to consider during capacity planning. In the near future I hope we can share results of recent benchmarking of Caché and ECP on some very large systems. As usual please let me know if you have any questions or anything to add through the comments. On twitter @murray_oldfield

#Best Practices #ECP #InterSystems Business Solutions and Architectures #InterSystems Data Platform Blog #System Administration #Caché #InterSystems IRIS #InterSystems IRIS for Health

Source URL: https://community.intersystems.com/post/data-platforms-and-performance-part-7-ecp-performance-scalability-and-availability