

# Re: Concurrently streaming a file to HttpResponseMessage and file IO

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-04/msg03834.html>

---

- *From:* Arne Vajhøj <[arne@xxxxxxxxxx](mailto:arne@xxxxxxxxxx)>
  - *Date:* Mon, 28 Apr 2008 22:39:54 -0400
- 

Anders Borum wrote:

I'm implementing support for disk based caching of binary resources (blobs) residing in a SQL database. This post is about choosing the right strategy.

Because of the web environment, there are potentially many concurrent requests to a resource. I would like to keep the application responsive (continue to serve requests for resources) while streaming resources to disk.

The case scenario is a number of concurrent requests (e.g. 10 or 20) asking for a resource (e.g. with a size of e.g. 32 MB) while the resource has not yet been cached on disk. In order to serve each request (in keeping the application responsive), an approach is to start streaming the resource from DB to the client request – and simultaneously queue a task to the threadpool that streams the resource to disk.

I'm thinking about implementing a producer / consumer pattern here; a producer creates tasks that the consumer picks up and starts streaming to disk.

Another approach would be to receive the request and check for the file existence (using a cache for quick lookups). If not cached, then check whether a "streaming register" contains information about the current file currently being streamed to disk. If not in the "streaming register", then queue a task to the thread pool (each worker thread is responsible for registering / unregistering the current streaming process).

Locking semantics should ensure that only a single thread is able to stream the same file to disk (e.g. it makes no sense that two threads are trying to stream the same file to disk in parallel).

I've already got all the major parts in place;

1. Authentication of requests to resources.
2. Managing http headers (result codes, mime types etc.).
3. Streaming large resources to disk from the DB (files are stored with a unique identifier (guid) plus a .cache extension).
4. Streaming large resources to a client response from the DB (using the chunk pattern).
5. Transmitting disk based resources to a client response (using IIS infrastructure for high performance).

Re: Concurrently streaming a file to HttpResponse and file IO

6. Scavenging of disk based resources not requested for a certain threshold.

I am a bit skeptical about the approach.

1) Is the disk cache really giving a significant performance improvement ?

(should be measured)

2) If yes – then why not just have all the files on disk instead of copying them out when actually used ?

3) If you want to proceed then I think the easiest approach would be to have the first request read from DB and write to both response and disk. Because then it will only need to be read once.

4) You need to consider how this solution will work in a cluster (local file caches and local repositories) would not scale very well.

5) Are you sure that the security provided by IIS serving files on disk are sufficient ?

Arne

.