Bottleneck Analysis for Cloud Transaction Architectures

Y.C. Tay
National University of Singapore

Abstract

At the SIGMOD 2010 conference, Kossman, Kraska and Loesing presented an experimental comparison of four cloud architectures for transaction processing. The paper concluded that "It is still unclear whether the observed results are an artifact of the level of maturity of the studied services or fundamental to the chosen architecture". This issue is addressed here via a theoretical analysis that focuses on the bottleneck in each architecture.

1 Introduction

The experimental comparison by Kossman, Kraska and Loesing [KKL] considers four architectures: **classic** (Fig.1), **partition** (Fig.2), **replicated** (Fig.3) and **dist.control** (Fig.4). The figures show — for each architecture — a brief description, the major issue, and one or two commercial examples.

The comparison considers transaction processing (OLTP), not business analytics (OLAP). It uses, as workload, the TPC-W benchmark that models an online bookstore by running an update-intensive mix of 14 transaction types.

The workload intensity is controlled by varying the number of emulated browsers (*EB*) that were run on EC2 machines in Amazon's cloud. The throughput comparison uses *WIPS* (web interactions per second) as metric.

The dataset had 10000 items that added up to 315MB of raw data and 1GB of index. (Image blobs were in a filesystem outside the database.) The database server is the bottleneck, except for **replicated** (where Azure used a high-end database server) and **dist.control** (S3).

Fig.5 from the experiments shows that the different architectures have distinctly different throughput behavior. In their conclusion, the authors asked if such differences were artifacts induced by the services, or fundamental to the architectures.

In the following, we present a bottleneck analysis to address the question posed by the authors.

2 Bottlenecks

The experimental comparison focuses on throughput limits, so we use bottleneck analysis [Tay] to study the asymptotic bound as EB increases.

Bottleneck analysis cannot model the throughput degradation seen in the **partition+replicated** architectures (SDB and AppEng); modeling such degradation would require an analysis of the consistency and reliability protocol.

Table 1 lists some of the variables used in the analysis.

2.1 Assumptions

(A1) Web/application servers and storage servers use similar (commodity) machines $\mathcal{M}$.

(A2) For **classic**, the database server uses a machine that is c times as fast as $\mathcal{M}$.

(A3) Trasaction arrivals are evenly distributed among the servers.

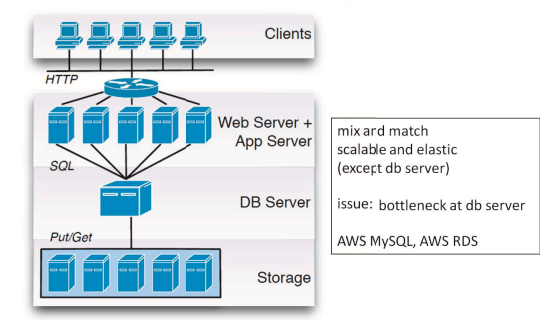

Figure 1: Classic Database Architecture

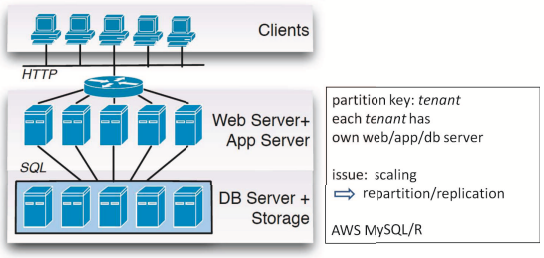

Figure 2: Partitioning

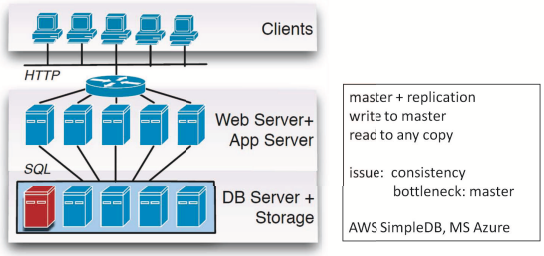

Figure 3: Replication

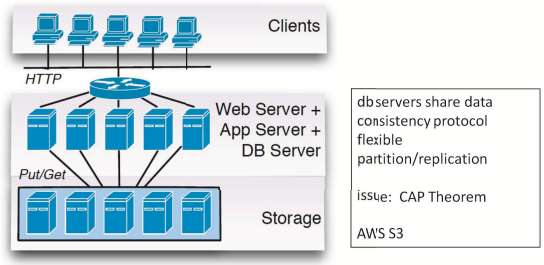

Figure 4: Distributed Control

2.2 Classic

By (A3), the transaction arrival rate at a web/application server is $\frac{\lambda}{N_{WA}}$; in steady state, the server utilization is $\frac{\lambda}{N_{WA}} D_{WA}$.

Utilization is at most 1, so we get an asymptotic bound

$$\lambda \leq \frac{N_{WA}}{D_{WA}}.$$

If λ exceeds this bound, the system cannot settle into a steady state: queues build up, buffers overflow, etc. Conversely, if this bound holds, then the system converges to a steady state where the throughput is λ. Similarly,

$$\lambda \leq \frac{N_{st}}{D_{st}}.$$

By (A2), the service demand at the database server in **classic** is $\frac{D_{db}}{c}$, so

$$\lambda \leq \frac{c}{D_{db}}.$$

Together, we get $\lambda \leq \lambda_{classic}$, where

$$\lambda_{classic} = \min\left\{\frac{N_{WA}}{D_{WA}}, \frac{c}{D_{db}}, \frac{N_{st}}{D_{st}}\right\}. \quad (1)$$

This is the saturation throughput and the limit on steady state transaction arrivals.

2.3 Partition

Similarly, the bound for **partition** is $\lambda \leq \lambda_{partition}$, where

$$\lambda_{partition} = \min\left\{\frac{N_{WA}}{D_{WA}}, \frac{N'_{st}}{D_{db} + D_{st}}\right\}. \quad (2)$$

2.4 Distributed

For **dist.control**, we have $\lambda \leq \lambda_{dist.control}$, where

$$\lambda_{dist.control} = \min\left\{\frac{N'_{WA}}{D_{db} + D_{WA}}, \frac{N_{st}}{D_{st}}\right\}. \quad (3)$$

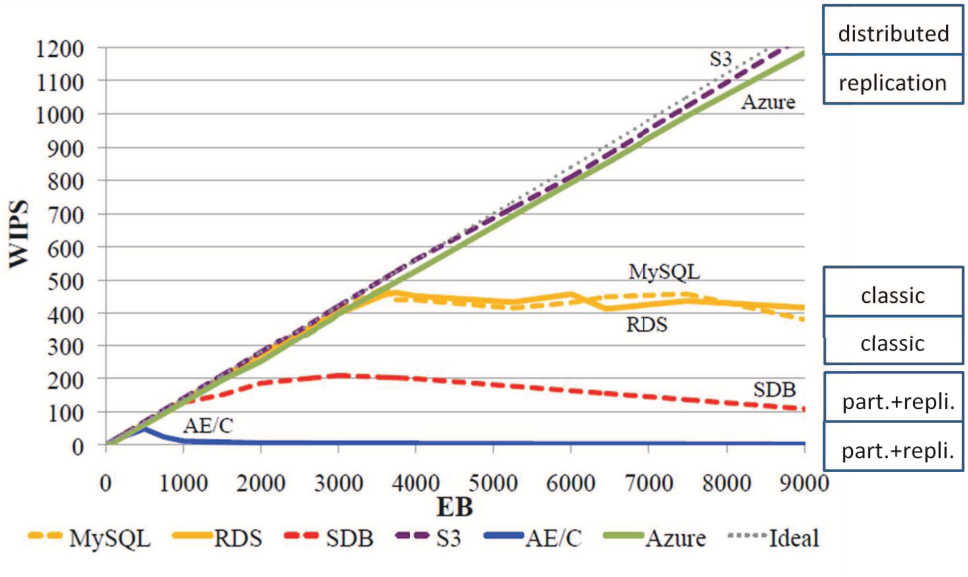

Figure 5: **Comparison of Architectures [WIPS]**

3 Analysis

Fig. 5 shows

- one **dist.control** architecture, namely Amazon's S3;
- one **replicated** architecture, namely Microsoft's Azure;
- two **classic** architectures, namely Amazon's MySQL and relational database service (RDS);
- two **partition+replicated** architectures, namely Amazon's SimpleDB (SDB) and Google AppEngine with MemCache (AE/C);

We now use the equations to analyze the throughput limits shown in the plot. (We omit Microsoft Azure, since we do not model **replicated**.)

3.1 Classic vs Partition

The two **partition+replicated** architectures perform worse than the **classic**. Is this inherent in the architectures?

If the bottleneck is in the database server(s), then

$$\begin{aligned} \lambda_{classic} &= \frac{c}{D_{db}} \\ \lambda_{partition} &= \frac{N'_{st}}{D_{db} + D_{st}} \\ \lambda_{dist.control} &= \frac{N'_{WA}}{D_{db} + D_{WA}} \ . \end{aligned} \tag{4}$$

In particular, $\lambda_{partition}$ (for SDB and AE/C) can be raised to match $\lambda_{classic}$ (for MySQL and RDS) in Fig. 5 if $\frac{N'_{st}}{D_{db}+D_{st}} = \frac{c}{D_{db}}$, i.e.

$$N'_{st} = (1 + \frac{D_{st}}{D_{db}})c. \tag{5}$$

3.2 Classic vs Distributed Control

In Fig. 5, the throughput for **dist.control** (Amazon's S3) eventually exceeds that for **classic**, and appears to scale linearly as EB increases. Since no system scales linearly forever. where is the limit for **dist.control**?

For the configuration in the experiments, p586 of the paper [KKL] says a medium EC2 machine

Model Parameters	
N_{WA}	number of web/application servers *without* co-located database servers
N'_{WA}	number of web/application servers *with* co-located database servers
N_{st}	number of storage servers *without* co-located database servers
N'_{st}	number of storage servers *with* co-located database servers
D_{WA}	service demand (seconds on $\mathcal{M}$) per transaction at web/application server
D_{st}	service demand (seconds on $\mathcal{M}$) per transaction at storage server
D_{db}	service demand (seconds on $\mathcal{M}$) per transaction at database server
c	ratio of database server speed in **classic** architecture to $\mathcal{M}$
λ	transaction arrival rate
EB	number of emulated TPC-W browsers
WIPS	web interactions per second

Table 1: **Glossary**

can support 1500EBs as web/application server and 900EBs as web/application/database server. This suggests

$$\begin{aligned} 1500 D_{WA} &= 900(D_{WA} + D_{db}) \\ \text{i.e. } D_{WA} &= \frac{900}{600} D_{db} = \frac{3}{2} D_{db} \end{aligned} \tag{6}$$

(so D_{WA} is actually larger than D_{db}).

Moreover, $N_{WA} = 6$ and $N'_{WA} = 10$; subsituting these and Eqn.(6) into Eqns.(1)–(3), we get

$$\begin{aligned} \lambda_{classic} &= \min\left\{\frac{4}{D_{db}}, \frac{c}{D_{db}}, \frac{N_{st}}{D_{st}}\right\} \\ \lambda_{partition} &= \min\left\{\frac{4}{D_{db}}, \frac{N'_{st}}{D_{db} + D_{st}}\right\} \\ \lambda_{dist.control} &= \min\left\{\frac{4}{D_{db}}, \frac{N_{st}}{D_{st}}\right\}. \end{aligned} \tag{7}$$

Thus, for $\lambda_{classic} = \lambda_{dist.control}$, it suffices that $c \geq 4$.

If $c = 1$ for $\lambda_{classic}$ in the MySQL and RDS experiments, this suggests that their throughputs can be raised 4 times to match $\lambda_{dist.control}$ for S3, i.e. S3 throughput in fact saturates at 1800WIPS.

4 Conclusion

Our analysis of the issue posed by Kossman, Kraska and Loesing shows that the throughput differences that they observed are not fundamental to the architectures. In particular,

- **partition** can match **classic** if there are enough storage servers (Eqn. (5));
- **dist.control** has a saturation throughput that **classic** can match if its database server were fast enough (Sec. 3.2).

These conclusions may err if our assumptions (Sec. 2.1) are wrong or if we misunderstood the paper (e.g. Eqn.(6)). However, bottleneck analysis is a powerful technique that requires minimal assumptions (e.g. we did not assume any particular arrival process or execution time distribution). We expect the analysis can be fine-tuned to remove any error and thus model the architectures more accurately.

5 References

[**KKL**] D. Kossmann, T. Kraska and S. Loesing, An evaluation of alternative architectures for transaction processing in the cloud, *Proc. SIGMOD*, 2010.
http://dl.acm.org/citation.cfm?id=1807231

[**Tay**] Y.C. Tay, *Analytical Performance Modeling for Computer Systems*, Morgan & Claypool (2010).
http://www.morganclaypool.com/toc/csl/2/1