

Scaling XML Using a Beowulf Cluster

John J Chelsom
University of Victoria

Jay H Chelsom
Abingdon School

Abstract

We describe a series of experiments which test the performance and scalability of an XML records system deployed on a Beowulf cluster of open source XML databases. Using the open source cityEHR health records system as an example, we first ran experiments to determine the feasibility and optimal size of database instances running on Raspberry Pi and low-cost Intel computers. We describe the implementation of a Data Access Layer for create, read, query and delete operations, using XForms submissions, which encapsulates all database access. We then present the results of testing the scalability and performance of this implementation on clusters of one to sixteen physical database nodes. We conclude that Beowulf clustering provides an effective and cost-efficient mechanism for scaling XML records systems.

Introduction

A Beowulf cluster is a network of commodity-based computers, controlled by a single server and designed to support high performance, parallel processing. The name derives from an experimental system built in the mid 1990's by Becker, Sterling et al [2.,7.] who likened their architecture to the legendary Scandinavian prince, Beowulf, immortalised in an epic Old English poem [4.]. In the poem, the unarmed Beowulf enlists the help of his companions to slay the monster Grendel, using his bare hands. Hence the analogy with using a large cluster of readily-available computers to achieve high performance computing tasks.

We have previously reported on experiments to scale an XML application using a network of clustered database nodes [3.]. Here we report on extensions to this work, to create a more general Data Access Layer for (DAL) for create, read, query and delete operations. The experiments are made using the cityEHR open source health records system. This is an XML application (XForms, REST, XQuery) created on an Enterprise Java platform of open source components, notably the Orbeon Forms application framework and the eXist native XML database.

Two specific design goals for the cityEHR system is that it should use the standard Orbeon and eXist packages, without modification, and that the system should deploy and scale 'out-of-the-box' in a way that can be accomplished by clinician users, without the intervention of IT specialists. Although cityEHR is an Enterprise Java application, it has no Java code itself – all code is XML (XForms/XHTML, XSLT, XML Pipelines and XQuery).

This design philosophy constrains the development of the capabilities for scaling the application, to the extent that the arbitrarily scalable, clustered database is implemented using:

- standard instances of the eXist database
- XForms code only
- dynamic discovery of deployed database nodes
- configuration through the web browser (XForms) user interface

As one of the most prominent, low-cost commodity computers, the Raspberry Pi has been used in a number of experimental Beowulf clusters, including those reported at Bolzano [1.], Glasgow [8.], Boise State [5.] and Texas A&M University [9.]. One key objective has been to evaluate the feasibility of the Raspberry Pi for hosting an instance of the eXist database and to measure the performance of the Raspberry Pi as part of the Beowulf cluster.

The overall objectives of our experiments, described in the following sections, have been to:

- assess the feasibility of commodity computers for running the eXist XML database
- determine the optimal size of database instances, as nodes of a Beowulf cluster
- develop a flexible and robust Data Access Layer, using XForms, which can scale to an arbitrary level
- compare the performance of the Data Access Layer on Beowulf clusters of Raspberry Pi and Intel PCs, ranging from one to sixteen database nodes
- extrapolate results to predict the extensibility of this approach to scaling XML using a Beowulf cluster.

Methods

Sizing a Single Node

The first experiments were undertaken to find the optimal configuration of a single database instance for three hardware types:

- Raspberry Pi, Model B 700 MHz single-core ARM1176JZF-S, 512Mb RAM
- Raspberry Pi 3, Model B 1.2 GHz 64-bit quad-core ARM Cortex-A53, 1Gb RAM
- Intel Core 2 Duo E7400 2.80 GHz processor, 4Gb RAM

The comparative performance ratings for these three processors on Linpack double precision arithmetic tests, in MFLOPS, are 42, 176 and 1577 [6.]

Comparison of the hardware platforms was made using an out-of-the-box installation of eXist 4.1, with its default configuration of 512Mb working memory. The database was loaded through the test record generation feature of cityEHR, using a template patient record of 34 XML documents, totalling 713Kb per record.

The write performance of the database was measured by generating batches of records from 1 to 500 patients; the results are shown in Table 1.

Batch size (records)	1	10	25	50	75	100	200	300	400	500
Final database size (records)	1	11	36	86	161	261	461	761	1161	1661
Raspberry Pi 2										
Total write time (secs)										
Write time / record (secs)										
Raspberry Pi 3										
Total write time (secs)	8.5	37	86	171	254	353	695	1029	1439	1879
Write time / record (secs)	8.50	3.70	3.44	3.42	3.39	3.53	3.48	3.43	3.60	3.76
Intel Dual Core										
Total write time (secs)										
Write time / record (secs)										

Table 1. Write time of batches from 1 to 500 records

Query performance of the database was measured using a sample XQuery to return all female patients in the database, which due to the data generation method is roughly half the total patients.

```
xquery version "1.0";
declare namespace cda="urn:hl7-org:v3";

/descendant::cda:value[@value eq 'Female']
```

```
[@extension eq '#ISO-13606:Element:Gender']  
/ancestor::cda:observation[cda:id/@extension eq '#ISO-13606:Entry:Demographics']  
/ancestor::cda:ClinicalDocument/descendant::cda:patientRole/cda:id
```

The query was run on databases ranging in size from 1161 to 10,000 records. The results are shown in Table 2.

X-Forms Data Access Layer

Implementation of the Beowulf cluster was made using the same XForms framework that underpins the cityEHR records system. In previous versions of the system, database transactions (create, read, query and delete) were performed using individual XForms submissions to the REST interface of the eXist database. For the Beowulf cluster implementation, a Data Access Layer (DAL) was introduced so that all database transactions are encapsulated in four specific submissions, one for each transaction type.

Introducing the Data Access Layer means that the database cluster can be configured independently, without impact on other XForms code. In particular, the data access layer allows configuration of:

- the physical nodes in the Beowulf cluster
- optimal capacity of each node
- logical configuration of nodes (independent of physical IP address)
- a single, logical database, independent of the logical or physical clustering
- replication of the database

The Data Access Layer introduces some overhead in database interaction and an important part of the evaluation has been to quantify the overall effect on performance.

Beowulf Cluster of Raspberry Pi 3

A Beowulf cluster of 16 nodes was implemented using the Raspberry Pi 3, Model B. Each node was loaded to the optimal capacity determined in the previous experiment for sizing a single node, to create a database of 160,000 records, consisting of a total of 5.44 million XML documents.

Beowulf Cluster of Intel Core 2 Duo E7400

An equivalent Beowulf cluster of 16 nodes was implemented using the Intel Core 2 Duo E7400 PCs.

Results and Analysis

Data Access Layer

The performance and scalability of the Data Access Layer was evaluated using a database of up to 100 logical database nodes, each of a maximum 10,000 records, for a total of up to 1 million records. For the purpose of this evaluation, each record consisted of a single XML document and the database was deployed on a single physical node.

Logical Nodes	1	5	10	15	20	30	40	50	60	70	80	90	100
Database size (records)	10000	50000	100000	150000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
Direct database access													
Write 10 records (sec)													
Read 1 document (sec)													
Query (sec)													
Data Access Layer													
Write 10 records (sec)													
Read 1 document (sec)													
Query (sec)													

Beowulf Cluster

Performance of the two Beowulf clusters was compared for write, read and query operations on clusters of 1 to 16 database nodes, where each logical node was deployed on a single physical node (so the number of physical and logical nodes was identical).

Database Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Database size (records)	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000	110000	120000	130000	140000	150000	160000
Database size (documents)	340000	680000	1020000	1360000	1700000	2040000	2380000	2720000	3060000	3400000	3740000	4080000	4420000	4760000	5100000	5440000
Raspberry Pi 3																
Read 1 document (sec)																
Query, 1 result (sec)																
Query, multiple results (sec)																
Intel Dual Core																
Read 1 document (sec)																
Query, 1 result (sec)																
Query, multiple results (sec)																

Conclusions

These experiments have shown that XML records systems, such as the cityEHR, can be scaled to millions of records and tens of millions of documents using a Beowulf cluster. There is some performance overhead in using the clustering approach but this is offset by the gain in scalability. Smaller scale systems can be configured with a single database node, or a small number of nodes, to reduce the performance overhead to a minimum but allowing for an increase in the number of database nodes should the number of records increase beyond the capacity of the original node(s).

References

1. Abrahamsson, P., Helmer, S., Phaphoom, N., Nicolodi, L., Preda, N., Miori, L., Angriman, M., Rikkila, J., Wang, X., Hamily, K. and Bugoloni, S., 2013, December. Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on* (Vol. 2, pp. 170-175). IEEE.
2. Becker, D.J., Sterling, T., Savarese, D., Dorband, J.E., Ranawak, U.A. and Packer, C.V., 1995, August. BEOWULF: A parallel workstation for scientific computation. In *Proceedings, International Conference on Parallel Processing* (Vol. 95, pp. 11-14).
3. Chelsom, John (2016) Scalability of an Open Source XML Database for Big Data. Proceedings of XML London, 2016.
4. Heaney, S., 2000. trans. Beowulf. *Beowulf: A Verse Translation*.

5. Kiepert, J., 2013. Creating a raspberry pi-based beowulf cluster. *Boise State*.
6. Longbottom, R., 2014. Linpack benchmark results on PCs.
<http://www.roylongbottom.org.uk/linpack%20results.htm>
7. Sterling, T., Becker, D.J., Berry, M.R., Savarese, D. and Reschke, C., 1996, April. Achieving a balanced low-cost architecture for mass storage management through multiple fast ethernet channels on the beowulf parallel workstation. In *Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International* (pp. 104-108). IEEE.
8. Tso, F.P., White, D.R., Jouet, S., Singer, J. and Pezaros, D.P., 2013, July. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. In *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on* (pp. 108-112). IEEE.
9. Wilcox, E., Jhunjhunwala, P., Gopavaram, K. and Herrera, J., 2015. *Pi-crust: a Raspberry Pi cluster implementation*. Technical report, Texas A&M University.