SIG Proceedings Paper in LaTeX Format[*]

Extended Abstract[†]

J. McFadden[‡]
Institute of Technology
Univ. of Washington: Tacoma
Tacoma, WA 98402
mcfaddja@uw.edu

Y. Tamta[§]
Institute of Technology
Univ. of Washington: Tacoma
Tacoma, WA 98402
yashaswitamta@gmail.com

J. N. Gandhi[¶]
Institute of Technology
Univ. of Washington: Tacoma
Tacoma, WA 98402
jugalg.uw.edu

ABSTRACT

Two different No-SQL database software packages will be implemented in and/or installed on several different types of systems. In turn, these systems will be deployed using several different types of platforms. This variety in software, systems, and platforms will allow different combinations of these to be compared based on metrics chosen in advance. In general, the metrics will include performance, setup, maintenance, and costs. These metrics will also be sufficiently detailed, precise, and consistent to a provide reliable assessment, as well as having sufficient scope to provide a viable comparison. and

CCS CONCEPTS

• **Computer systems organization → Cloud based systems**; *Software as a Service*; *Containers Service*; *Virtualization*; • **Database systems** → NoSQL; • **Virtualization methods** → Containers; • **Cloud Systems** → Perfomance; Cost;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
J. McFadden, Y. Tamta, and J. N. Gandhi. 1997. SIG Proceedings Paper in LaTeX Format. In *Proceedings of TCSS 562: Cloud Computing Term Project, Tacoma, WA USA, April 2017 (TCSS 562: Spring 2017),* **??** pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

The software/systems chosen for comparison in this project are two different NoSQL database system. These systems will be deployed/run/operated in several different ways. These include *SaaS*[1] implementations, *containerized* implementations, and *native installations.* The goal of the project is to understand the performance characteristics of each deployment method ***and*** to quantify the costs of each deployment method. These costs will be calculated based on the hourly cost to operate, the initial time & costs required for setup, and the maintenance requirement of a deployment. Additionally, performance of the systems and deployments will be measured using the time required to carry out various database operations, under a set of several different conditions, as well as the CPU, memory, and network loads imposed by the various deployments under the same set of conditions.

[*]Produces the permission block, and copyright information
[†]The full version of the author's guide is available as `acmart.pdf` document
[‡]
[§]
[¶]
[1]**SaaS :** Software as a service.

2 SYSTEMS AND PLATFORMS

We will be using two NoSQL database software packages. The first software package is **DynamoDB** from Amazon Web Services (*AWS*), while the second software package will be **Cassandra**, an open-source NoSQL database software package. These software packages will be deployed using several different systems and platforms, as described below.

2.1 Systems

This project will run the software packages on three different systems (*or types of systems*). We have chosen systems which range from hosted *SaaS* through various degrees of virtualization and then all the way to non-virtualized machines. These systems are as follows

A): ***SaaS***

B): **Running inside a *Docker* container**

C): **Running on a dedicated machine**

These four systems will be deployed using several different platforms which we will describe in the next part of this section; however, before proceeding to that, we will give a through description of each system listed above.

*2.1.1 SaaS (code: **A**).* In this system paradigm, a particular software package is developed and maintained with the intent of being used by a provider as selling *Software as a Service.* As far as system configuration on our end (*the system-administrator/developer*), paradigm only requires that we select a platform provider and collected the database data so that it can eventually be prepared for deployment.

2.1.2 Running inside a Docker *container (code: **B**).* For the purposes for this paper and the ensuing project, we will only use

containers which are created using and for use in *Docker*. This choice was made because of *Docker*'s ubiquity, wide availability, and the ease of access to cloud services designed for and solely dedicated to the deployment of *Docker* containers. This implementation paradigm requires that a *Docker* container be created after which, the software package in question be installed into the container so that it can be run on the operating system employed in the container (*the OS*[2] *was chosen at the time of container creation*). Once the software has been installed, any information for and/or data required by it is either loaded into the container or, in the case of data *only*, the container is "pointed at" the location of the data required by the software. Finally, we must note that, as far as this paper and project are concerned, the use of a *Docker* container as a means of implementation does not imply the use of any particular platform for running the containers, or even a deployment of the *Docker* software/system itself.

*2.1.3 Running on a dedicated machine (code: **C**) .* The third system paradigm involves acquiring and configuring a dedicated machine which can be either an actual, physical machine or a virtual machine. In either case, the machine will have performance specifications and and run an OS appropriate for our planed use. When setting up one of these machines, after configuration and setup of physical/virtual system, the selected OS will be installed and configured for use. Once it has been verified that the OS has been properly installed and configured (*i.e. running stably*) we will install the software selected for testing on that instance, along with any other software or system components required to run that software. After configuring the installed software, the machine will be prepared and configured for use as a server running said software. The final step of this paradigm is preparing the software and system combination for final deployment to the selected platform.

2.2 Platforms

We have chosen four different platforms on which to deploy our systems. The chosen platforms span the range of cloud service paradigms from *SaaS* to *PaaS*[3] to *IaaS*[4]. We list the three platforms, along with two variations on one of the platforms, below

1): (An) AWS Software Service (*SaaS***)**

2): Containers (using *Docker***) running on**

- **i):** ***AWS's Container Service*** **(***PaaS***)**
- **ii):** ***AWS EC2 VM with Docker run-time*** **(**hybrid *Pass/IaaS***)**

3): AWS EC2 VMs running the software in Linux (*IaaS***)**

4): A dedicated, non-virtualized server (*server***)**

In the next section, we list which systems will run each software package, along with a explanation why each software-system pairing was chosen. Additionally, we will describe which platforms will be used to deploy each system and why those deployment choices were made. Before proceeding to that, and as with our list of system, we will describe each platform thoroughly.

*2.2.1 AWS Software Service (code: **1**).* Our first platform we will use for the deployment of some systems is AWS's Software Service platform which provides various software packages which run as a service. This means that Administrator do not have to configure and secure a server and OS on which to run the software, nor do they have to install and manage the running of the software on said server. Instead, all a System Administrator must do is create an instance of the desired Software Service, with the configuration required for the application at hand. Once the instance has been created, it is loaded with any information required and/or data to be used, after which it is immediately available for use by users and other clients.

*2.2.2 Containers (code: **2_**).* [5]

The next platform we have chosen involves the deployment of the previously described *Docker* containers. We will use two different platforms that implement the *Docker* engine which is actually responsible for running the containers. These "*Docker* Platforms" are

- [*AWS's Container Service (code:* **2i***)*]**:** In this version of the second platform, we will deploy our containers by using AWS's Container Service. When deploying to this platform, an Administrator first creates an appropriately configured instance of the Container Service. Upon the successful creation and launch of the Container Service instance in question, the Administrator simply uploads the container being deployed to the instance and, if a data-source external to the container is required, uploads the required data to the location specified during the creation of the container. Once the container has been uploaded to the instance (*and any required data has been uploaded to the appropriate location*), the Administrator tests the instance to ensure it is functioning as expected. In the last step, the Administrator configures the instance to properly connect with users and/or clients before finally granting them access to the deployed container.
- [*AWS EC2 VM with Docker run-time (code:* **2ii***)*]**:** The other version of the second deployment platform involves deploying our containers using a *Docker* run-time/engine running in a Linux machine being hosted on AWS's EC2 Virtualization Platform. To deploy via this platform, an Administrator begins by creating an EC2 VM instance with performance characteristics necessary for running the container inside the Linux version of *Docker* run-time. Once the EC2 VM instance has been created and successfully launched, the Administrator will install and configure an appropriate version of Linux, followed by installing the *Docker* run-time and any required support software (*such as software for accessing remotely stored data*). After configuring the

[2]**OS:** Operating System
[3]**PaaS :** Platform as a service.
[4]**IaaS :** Infrastructure as a service.

[5]In the code **2_**, we use the "_" after the 2 is a place-holder for an additional code-letter. This is due to the fact that two different versions of and approaches to this platform (*container deployment*) are going to be used.

installation of the *Docker* run-time (*and any support software*), the Administrator will upload the container using *Docker*.
Additionally, if a data-source external to the container is employed, up to two additional steps are required and based on whether the data is local or remote to the VM instance. In the case the data-source is local to the VM instance, the Administrator must also configure appropriate storage on the instance, including installing and configuring any necessary software, after which the required data is uploaded to the storage location. Similarly, if the data-source is external to the VM instance, the Administrator must also install and configure the software required for accessing the remote location (*NFS, SAN, Samba, EBS, etc*), followed by configuring the system to access the remote location and the data stored within. Once these steps are complete, the Administrator will be able configure the container to run on the VM instance, as well as configure
the OS to allow the container to properly run. These configurations involve ensuring that the container has access to the necessary networks, addresses, and ports and that any local and/or remote resources are both appropriately mounted to the container (*container configuration*) and accessible to the container (*OS configuration*). With these configurations complete, the Administrator is able to launch the container and begin testing it to ensure it functions as expected. Once the container has been determined to be properly functioning, the Administrator completes any configuration of the EC2 Instance, the VM itself (*OS*), and the container itself which might be necessary for proper user and/or client access of the container. This final configuration is followed, finally, by allowing users and clients to access the container.

*2.2.3 AWS EC2 VMs running the software in Linux (code: **3**).* Our third deployment platform paradigm also relies on use of AWS's EC2 VM service; however, instead of installing using containers as in **2ii**, this paradigm has the software to be tested natively and directly installed on a Linux OS being run on an AWS EC2 VM instance. As with the previous paradigm involving an AWS EC2 VM the Administrator begins by creating an EC2 VM instance, launching it, and installing an appropriate version of the Linux operating system, with the only difference being the determination of necessary performance characteristics base on natively running the software, and any supporting software, in Linux. Following the installation and configuration of the Linux OS, the Administrator will natively install and then configure the software to be tested, along with any other software required support software (*i.e. remote data access*). These installations are followed by creating and configuring any required local storage space on the instance, to include installing and configuring any software necessary for doing this. In the case where remote storage be used instead of or in addition to local storage, the Administrator must also install and configure the software required for accessing the remote location, followed by configuring the system to access the remote location and the data stored within. Once these data storage/access steps have been completed, the Administrator uploads the required data to the appropriate location or locations, after which they configure both the software being tested, so that it can locate these data-sources, and the OS, so the software can access both the data-sources and any required networks, services, or ports. As before, this configuration step is followed first by testing to ensure the software is accessible and works as desired, and then by the final step, allowing users and clients to access the container.

*2.2.4 A dedicated, non-virtualized server (code: **4**).* The last deployment platform we have chosen is a dedicated, non-virtualized server. These servers will run Linux as their OS and will be dedicated to running and serving the software to be tested.

3 DEPLOYMENT

The deployment strategy will be described in two parts which can be briefly described as "*what systems will be used to run and/or implement each software package*" and "*which platforms will be used to deploy each of these systems*". Specifically, the first part of the description for our deployment strategy details which systems will be used to run each software package (*DynamoDB or Cassandra*), along with how the each system will implement and run the software. This will be complemented by the second part of the description, which will detail what platforms will be used for deploying each of the systems, along with why each platform was chosen to deploy are particular software/system combination.

3.1 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

3.1.1 Inline (In-text) Equations. A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin . . . \end` construction or with the short form `$ . . . $`. You can use any of the symbols and structures, from α to ω, available in LaTeX [?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

3.1.2 Display Equations. A numbered display equation—one set off by vertical space from the text and centered horizontally—is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty}x_i=\int_{0}^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

3.2 Citations

Citations to articles [? ? ? ?], conference proceedings [?] or maybe books [? ?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide* by Lamport [?].

This article shows only the plainest form of the citation command, using `\cite`.

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

Figure 1: A sample black and white graphic.

3.3 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table **??** is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table **??** is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

It is strongly recommended to use the package booktabs [**?**] and follow its main principles of typography with respect to tables:

(1) Never, ever use vertical rules.
(2) Never use double rules.

It is also a good idea not to overuse horizontal rules.

3.4 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of `.eps` files to be displayable with LaTeX. If you work with pdfLaTeX, use files in the `.pdf` format. Note that most modern TeX systems will convert `.eps` to `.pdf` for you on the fly. More details on each of these are found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. And don't forget to end the environment with **figure***, not **figure**!

Figure 2: A sample black and white graphic that has been resized with the `includegraphics` command.

3.5 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. ACM uses two types of these constructs: theorem-like and definition-like.

Here is a theorem:

THEOREM 3.1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)\,dt = G(b) - G(a).$$

Here is a definition:

Definition 3.2. If z is irrational, then by e^z we mean the unique number that has logarithm z:

$$\log e^z = z.$$

The pre-defined theorem-like constructs are **theorem**, **conjecture**, **proposition**, **lemma** and **corollary**. The pre-defined definition-like constructs are **example** and **definition**. You can add your own constructs using the *amsthm* interface [**?**]. The styles used in the `\theoremstyle` command are **acmplain** and **acmdefinition**.

Another construct is **proof**, for example,

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c} \left[gx \cdot \frac{f(x)}{g(x)} \right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. □

4 CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

Table 2: Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables

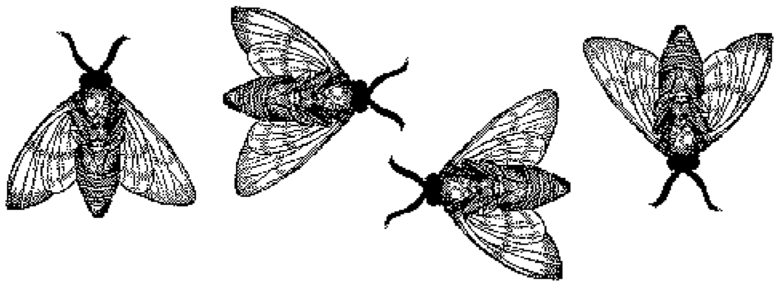

Figure 3: A sample black and white graphic that needs to span two columns of text.

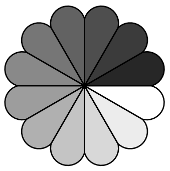

Figure 4: A sample black and white graphic that has been resized with the `includegraphics` command.

A HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e., the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters.

A.2.2 Math Equations.

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations.

A.2.4 Tables.

A.2.5 Figures.

A.2.6 Theorem-like Constructs.

A Caveat for the TEX Expert.

A.3 Conclusions

A.4 References

Generated by bibtex from your `.bib` file. Run latex, then bibtex, then latex twice (to resolve references) to create the `.bbl` file. Insert that `.bbl` file into the `.tex` source file and comment out the command `\thebibliography`.

B MORE HELP FOR THE HARDY

Of course, reading the source code is always useful. The file `acmart.pdf` contains both the user guide and the commented code.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Yuhua Li for providing the matlab code of the *BEPS* method.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304 and Young Scientsts' Support Program (http://www.nnsf.cn/youngscientsts).