Accommodate Apache YARN to long-lived services

Huiyi Li, Shanghai Jiaotong University

***Abstract*— We accommodate the Apache YARN (Yet Another Resource Negotiator) to long-lived services to complete its**

These instructions give you guidelines for preparing papers for the ICSGCE/IEEE conference. Use this document as a template by using Microsoft *Word* 6.0 or later. Otherwise, use this document as an instruction set. Please use this document as a "template" to prepare your manuscript. For submission guidelines, follow the instructions on paper submission on http://www.icsgce.com/. Do not delete the blank line immediately above the abstract; it sets the footnote at the bottom of this column.

***Keywords*-Component; formatting; style; styling; insert**

I. Introduction

Large-scale cluster scheduling/management systems [1, 2, 3, 4, 5] are becoming a dominant platform for a variety of applications and services. These complex "cloud" systems often run on clusters of thousands of unreliable commodity machines and mush handle all kinds of failures, schedule and deployment requirements while achieving the maximum resource utilization.

These systems all have two similar interests: improve cluster resource utilization and achieve auto deployment of service. In big data era, our data center usually runs multiple types of applications and servers simultaneously. Such as offline/stream/iterative computing job, web server, database server. Typically we dedicate a unique cluster to each type of services, but due to different resource requirements of different types of services, the resource utilization rate may vary differently. Some cluster may runs at a full capacity and some not. Therefore, resource management systems such as YARN, Mesos, Borg, Omega, Corona, Torca are designed to address this problem.

We take YARN to a full analysis and argue that YARN currently excels in handling batch jobs without interacting with outside system or clients. It takes a job and determines where the components such as Mapper and Reducer would live and deploy those components to different nodes to do their work. But when a long-lived services such as HBase, Storm, Spark appears, YARN has problem of telling the outside system where those services exists and how to interact with them. When a node becomes unavailable, YARN simply reschedule the component on that node to another one, this was feasible when dealing with map-reduce jobs where all components of the job are stateless and do not need to interact with each other directly. Take Storm as an example, when Nimbus node fails, YARN just reschedules it to another node, however all the other components of Storm needs to interact with Nimbus node, and since its relocated "randomly", there is no way to find out where the Nimbus node resides. Another problem with running long-lived service in YARN is that those services typically would generate a large amount of logs, and those logs are important to tracking the system state or even for debugging purposes, and YARN need to provide a way to aggregate those logs and make them easily accessible or even provide the ability to purge those logs while no longer needed. Additionally, upgrading a node in a YARN cluster can be a disruptive or lengthy process. If the node manager is taken down then all active containers on that node are killed. This is disruptive to long-lived services running on YARN.

In this paper, we propose a solution to the service registration, rolling upgrade, node relocation problem that YARN had and accommodate YARN to the long-lived services.

This paper is structured as follows. First, we discuss our solution and its effectiveness (§2). Next, we describe the design (§3) and implementation (§4). We then measure the effectiveness and performance of our solution (§5), discuss related work (§6), and conclude (§7).

II. Extended Motivation

A. *Service registration*

Service registration and discovery is a long-standing problem in distributed computing, dating back to Xerox's Grapevine Service. Apache YARN allows applications to run on the Hadoop cluster. Some of these are batch jobs or queries that can be managed via YARN's existing API using its jobId property. In addition YARN can deploy long-lived service instances such as an Apache HBase cluster or a pool of Apache Tomcat web servers. These services are deployed by YARN according to their individual component requirements and the server availability. These service instances need to be discovered by clients, traditionally their IP added is registered in DNS or in some configuration file – but that is not feasible in YARN-deployed applications while neither the hostname nor network ports can be known in advance.

As a result, there is no easy way for clients to interact with dynamically deployed long-lived services.

A rudimentary registry was supported by YARN which allows YARN Application Masters to register a web URL and an IPC address. However, this is not sufficient for many long-lived services since it does not allow other endpoints to be registered – such as REST URLs, or the endpoints of tasks that the Application Master executes. Furthermore, the information that can be registered is mapped to the YARN application instance – a unique instance ID that changes whenever a YARN application is started. This makes it impossible to resolve binding information via a static reference to a named service, or to even probe for the existence of a service instance which is not currently available.

We have four key requirements for addressing this problem which our proposed solution satisfies: 1. YARN deployed service instances should be able to register their bindings and be discovered by clients and the binding must be upgradable if the service moves or in case of a HA fail over; 2. A service instance must be able to publish a variety of endpoints: RPC, REST, others; 3. Our registry service must be highly available and can scale to a large cluster; 4. Supports remote access even on clusters which are only reachable via Apache Knox [6], or hosts cloud environments.

B. Log management

Running long-lived services inside a YARN cluster will certainly result to an enormous amount of service logs. These logs are mostly not important when they have existed for a certain period of time. Currently YARN uses HDFS delegation token to deal with log management when the job is finished. HDFS delegation tokens have a maximum life time. Tokens submitted to the Resource Manager when the Application master is launched will be renewed by the Resource Manager until the application have finished aggregating. It is troublesome when the Resource Manager take longer than the maximum life time to aggregate logs since YARN has not provided a way to renew its token. Also, there is no way for long-lived services to choose when and how they want to deal with the logs.

We have four key requirements for addressing this problem which our proposed solution satisfies: 1. All YARN-deployed applications should have the ability to choose when and how they want to deal with their logs; 2. There should be a way to give applications demanded accessibility so that they would have the ability to deal with large amount of logs; 3. The configuration should be pluggable and configurable during the lifecycle of long-lived services.

C. Rolling upgrade

Upgrading all of the nodes in a cluster for a rolling upgrade can be a very disruptive or lengthy process. If a Node Manager was taken down, then all active containers on that node would be killed. This is disruptive to jobs with long-running tasks or long-lived services. There have been discussions about decommission one node for rolling upgrade or other maintenance issues. However with long-lived services it can take a very long time to decommission for nodes, as we not only have to wait for the active containers to complete but also active applications in general (e.g. node still has to serve up map task data after map task completes, so auxiliary services can have responsibilities beyond the active containers). Performing a rolling upgrade on a large cluster will take a very long time if we need to wait for a clean drain-decommission of each node.

Therefore, it would be nice if the Node Manager supports a node where it could be restarted and recover state. We could then bounce the Node Manager to an updated version without losing containers and with minimal impact to jobs running on the cluster, and the time to perform a rolling upgrade of a large cluster would no longer be tied to the running time of applications currently active on the cluster.

III. Design and Implementation

In this section, we describe our general approach to designing and implementing YARN to accommodate for long-lived services.

A. Node models

Like all similar resource management systems, YARN clusters consist of a single Resource Manager (RM) node and multiple node manager nodes. The resource manager is the master that arbitrates all the available cluster resources and thus helps manage the distributed applications running on the YARN cluster. It works together with the per-node Node Manager (NM) and the per-application Application Master (AM). NMs take instructions from the RM and manage resources available on a single node which includes keeping up-to-date with the RM, overseeing containers' life-cycle management; monitoring resource usage (memory, CPU) of individual containers, tracking node-health, log's management and auxiliary services which may be exploited by different YARN applications. AMs are responsible for negotiating resources with the RM and for working with the NM to start the containers.

In conclusion, the RM is primarily limited to scheduling i.e. only arbitrating available resources in the system among the competing applications and not concerning itself with per-application state management. The NM is primarily limited to managing abstract containers i.e. only processes corresponding to a container and not concerning itself with per-application state management like MapReduce tasks.

B. Architecture

To accommodate YARN to long-lived services, we propose a base registry service that binds string-names to records describing service and component instances and provide a pluggable and configurable log management API, finally we propose a specific method for rolling upgrade the YARN cluster.

Figure 1 illustrates the overall architecture and workflow we use to solve the service registry problem. We plan to use Zookeeper [7] as the base name service since it supports

many of our required properties, then we pick a part of the Zookeeper namespace to be the root of the service registry (e.g. “<ZK-ROOT>/serviceRegistryRoot”).

Before we dive into details of our design, we need to address several basic concepts. Service is a potentially distributed application deployed in – or reachable from – a Apache YARN cluster (e.g. Apache Storm, Apache Spark). A Service Record is a record in the registry describing a service instance or a component instance. A component is a distributed element of a service (e.g. Storm Nimbus node). Endpoint is one means of binding with a service instance or a component instance (e.g. a Java JMX port on a region server). Endpoints can be both external – for use by programs other than service itself, and internal – for connecting components within the service instance.

We register every Service by binding a path to a value called a Service Record. The paths are hierarchical and use “/” as the root and “/” as the separator. All path elements must match that of a lower-case entry in a hostname path as defined in RFC1123 [8]. This makes the naming convention match that of DNS so that we have the option of accessing the namespace via DNS protocol. Every Service Record is registered as persistent znodes. This ensures that the record remains present during planned and unplanned outages of the service, on the assumption that client code is resilient to transient outages.

Each service instance’s Service Record lists the Endpoints for its various protocols exported by that service instance. For each protocol Endpoint it must contain:

a. The protocol name including: WEB, REST, IPC etc.
b. Its address: the specific details used to locate this Endpoint
c. Its addressType. This is the format of the binding string (URL, Zookeeper path, hostname/port pair)
d. The api. This is the API offered by the endpoint, and is application specific. (e.g. jmx, www)

Figure 2 is an example of the Service Record.

```
{
   "registrationTime": 1408638082444,
   "id": "application_001",
   "description": "tomcat-based web application",
   "persistence": "0",
   "external": [ {
      "api": "www",
      "addressType": "uri",
      "protocolType": "REST",
      "addresses": [
         { "http://tomcat1" },
         { "http://tomcat2" }
      ]
   } ],
   "internal": []
}
```

Second, Service Records are registered as persistent znodes. This ensures that the record remains present during planned and unplanned outages of the service, on the assumption that client code is resilient to transient outages. Third, each service instance’s service record lists the endpoints for its various protocols exported by that service instance.

IV. Implementation

In this section, we describe the specific details of our system.

V. Evaluation

VI. Related work

Each figure and table should have a caption to concisely and intelligibly illustrate the contents of it. Figures/tables may be worked into the text or placed at the end of the manuscript. To conserve space in the publication, most figures/tables are reduced to single-column width if possible. This may result in as much as a 4:1 reduction from the original. Therefore, figures/tables should be kept to a minimum in original and be easily viewed on published pages. Large figures and tables may span both columns.

In the finalized sizes of figures/tables, authors are advised to make sure that (see Fig. 1):

- All images/photographs will be published in black-and-white, so do not describe any of images/photographs with words such as red line, blue area, etc.
- Graphing figures are recommended to generate in gray curves because some color lines will be not legible in black-and-white.
- Lines in the figures are in 0.75 pounds and arrows in the minimum.
- Mathematical expressions (variables) appearing in figures should be in the same styles as in texts (see Section III).
- Trigram tables are suggested, as in Table 1, the first and the last lines are double lines and the 2nd line is in 0.75 pounds.
- Texts in figures are approximately 8pt.
- Captions of figures and tables are approximately 9pt.
- Place figure captions below the figures, as in Fig. 1.
- Place table titles above the tables, as in Table 1.

The figures and tables are recommended to insert in your document after the text actually exists. **Please do not include captions as part of the figures. Do not put captions in “text boxes” linked to the figures.** Use the abbreviation “Fig.” even at the beginning of a sentence. Do not abbreviate “Tab.”. Tables are numbered with Arabic numerals.

Table 1: The arrangement of channels

Channels	Group 1	Group 2	⋯	Group *c*
Main channel	Channel 1	Channel 2	⋯	Channel *c*

...	...	...	...	...
Assistant channel	Channel 2	Channel 3	⋯	Channel 1

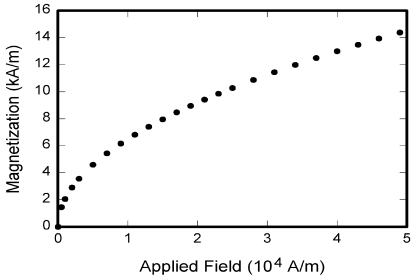

Fig. 1. Magnetization as a function of applied field. Note that "Fig." is abbreviated. There is a period after the figure number, followed by two spaces. It is good practice to explain the significance of the figure in the caption. If your figure has two parts, include the labels "(a)" and "(b)" below the corresponding part of the figure. Then the figure caption should be "The significance of the figure: (a) the significance of (a) and (b) the significance of (b)"

Figure axis labels are often a source of confusion. Use words rather than symbols. As an example, write the quantity "Magnetization," or "Magnetization *M*," not just "*M*." Put units in parentheses. Do not label axes only with units. As in Fig. 1, for example, write "Magnetization (A/m)" or "Magnetization ($A \cdot m^{-1}$)," not just "A/m." Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)," not "Temperature/K."

Multipliers can be especially confusing. Write "Magnetization (kA/m)" or "Magnetization (10^3 A/m)." Do not write "Magnetization (A/m)×1000" because the reader would not know whether the top axis label in Fig. 1 meant 16000 A/m or 0.016 A/m.

VII. Helpful Hints

Essentially, academic paper writing is as a form of problem-solving in which the writer, or the author, faces two main tasks: a) generating his academic ideas in language, and b) composing these ideas into a written structure to meet the need of readers and the requirements of the journal.

Generally speaking, writing a good paper in English requires the mastery of various skills. It requires language basis, grammatical accuracy and readability, so that relationship between words and sentences are clear, and understanding between reader and writer is made easier. Additionally, it requires vocabulary appropriate to the subject matter and to the level and tone of the paper. Finally, of more importance, writing a good academic paper requires a careful and well-planned structuring of ideas.

However, this Template is incapable to include everything you need to know to be a better writer. Given here are some useful language hints that should be an important part of resources for your paper writing.

A. *Formal Usages*

- Use one space after periods and colons.
- Hyphenate complex modifiers: "zero-field-cooled magnetization."
- Prefixes such as "non," "sub," "micro," "multi," and ""ultra" are not independent words; they should be joined to the words they modify, usually without a hyphen.
- Avoid dangling participles, such as, "Using (1), the potential was calculated." [It is not clear who or what used (1).] Write instead, "The potential was calculated by using (1)," or "Using (1), we calculated the potential."
- A parenthetical statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- Avoid contractions; for example, write "do not" instead of "don't." The serial comma is preferred: "A, B, and C" instead of "A, B and C."

B. *Some Common Mistakes*

- The word "data" is plural, not singular.
- The word "alternatively" is preferred to the word "alternately" (unless you really mean something that alternates).
- Use the word "whereas" instead of "while" (unless you are referring to simultaneous events).
- Do not use the word "issue" or "question" as a euphemism for "problem."
- Be aware of the different meanings of the homophones "affect" (usually a verb) and "effect" (usually a noun), "complement" and "compliment," "discreet" and "discrete," "principal" (e.g., "principal investigator") and "principle" (e.g., "principle of measurement"). Do not confuse "imply" and "infer."
- There is no period after the "et" in the Latin abbreviation "*et al.*" (It is also italicized).
- The abbreviation "i.e.," means "that is," and the abbreviation "e.g.," means "for example" (these abbreviations are not italicized).

C. *Abbreviations and Acronyms*

Define abbreviations and acronyms the first time they are used in the text, even after they have already been defined in the abstract. Abbreviations such as TCP/IP, ac, and dc do not have to be defined. Do not use abbreviations in the title unless they are unavoidable.

The abbreviation for "seconds" is "s," not "sec."

D. *Units*

Use SI not CGS as primary units. Avoid combining SI and CGS units. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

- Use the center dot to separate compound units, e.g., "$A \cdot m^2$."
- Indicate sample dimensions as "0.1 cm × 0.2 cm," not "0.1×0.2 cm^2."
- When expressing a range of values, write "7 to 9" or "7-9", not "7~9".

Remember that an excellent academic paper needs to be composed by authors in good language! Undecipherable English is a valid reason for rejection! If your native language is not English, please get a colleague good at English or a native English-speaker to proofread your paper.

VIII. References and Citations

Number citations consecutively in square brackets [1]. The sentence punctuation follows the brackets [2]. Multiple references [2], [3] are each numbered with separate brackets [1]–[3]. When citing a section in a book, please give the relevant page numbers [2]. In sentences, refer simply to the reference number, as in [3]. Do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] shows ... .” The conference cannot accept footnotes in its document; therefore, type the reference list at the end of the paper using the “References” style

Please note that the references at the end of this document are in the preferred referencing style. Give all authors’ names; do not use “*et al*.” unless there are six authors or more. Use a space after authors' initials. Papers that have not been published should be cited as “unpublished” [4]. Papers that have been submitted for publication should be cited as “submitted for publication” [5]. Papers that have been accepted for publication, but not yet specified for an issue should be cited as “to be published” [6]. Please give affiliations and addresses for private communications [7].

Capitalize only the first word in a paper title, except for proper nouns and element symbols. For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [8].

IX. Conclusion

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

X. Appendix

Appendixes, if needed, appear before the acknowledgment.

XI. Acknowledgment

Use the singular heading even if you have many acknowledgments. Avoid expressions such as “One of us (S.B.A.) would like to thank ... .” Instead, write “F. A. Author thanks ... .” **Sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page**.

References

[1] Vavilapalli, Vinod Kumar, et al. "Apache hadoop yarn: Yet another resource negotiator." Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013.

[2] Hindman, Benjamin, et al. "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center." NSDI. Vol. 11. 2011..

[3] Schwarzkopf, Malte, et al. "Omega: flexible, scalable schedulers for large compute clusters." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.

[4] Verma, Abhishek, et al. "Large-scale cluster management at Google with Borg." Proceedings of the Tenth European Conference on Computer Systems. ACM, 2015.

[5] Corona facebook https://github.com/facebookarchive/hadoop-20/tree/master/src/contrib/corona.

[6] Sharma P P, Navdeti C P. Securing Big Data Hadoop: A Review of Security Issues, Threats and Solution[J]. IJCSIT) International Journal of Computer Science and Information Technologies, 2014, 5(2).

[7] Hunt, Patrick, et al. "ZooKeeper: Wait-free Coordination for Internet-scale Systems." USENIX Annual Technical Conference. Vol. 8. 2010.

[8] RFC1123: http://www.ietf.org/rfc/rfc1123.txt

[9] M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.

[10] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility (Periodical style),” *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34–39, Jan. 1959.

[11] S. Chen, B. Mulgrew, and P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis function networks,” *IEEE Trans. Neural Networks*, vol. 4, pp. 570–578, July 1993.

[12] R. W. Lucky, “Automatic equalization for digital communication,” *Bell Syst. Tech. J.*, vol. 44, no. 4, pp. 547–588, Apr. 1965.

[13] S. P. Bingulac, “On the compatibility of adaptive controllers (Published Conference Proceedings style),” in *Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*, New York, 1994, pp. 8–16.

[14] G. R. Faulhaber, “Design of service systems with priority reservation,” in *Conf. Rec. 1995 IEEE Int. Conf. Communications*, pp. 3–8.

[15] W. D. Doyle, “Magnetization reversal in films with biaxial anisotropy,” in *1987 Proc. INTERMAG Conf.*, pp. 2.2-1–2.2-6.

[16] G. W. Juette and L. E. Zeffanella, “Radio noise currents n short sections on bundle conductors (Presented Conference Paper style),” presented at the IEEE Summer power Meeting, Dallas, TX, June 22–27, 1990, Paper 90 SM 690-0 PWRS.

[17] J. G. Kreifeldt, “An analysis of surface-detected EMG as an amplitude-modulated noise,” presented at the 1989 Int. Conf. Medicine and Biological Engineering, Chicago, IL.

[18] J. Williams, “Narrow-band analyzer (Thesis or Dissertation style),” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.

[19] N. Kawasaki, “Parametric study of thermal and chemical nonequilibrium nozzle flow,” M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[20] J. P. Wilkinson, “Nonlinear resonant circuit devices (Patent style),” U.S. Patent 3 624 12, July 16, 1990.

[21] *IEEE Criteria for Class IE Electric Systems* (Standards style), IEEE Standard 308, 1969.

[22] *Letter Symbols for Quantities*, ANSI Standard Y10.5-1968.

[23] R. E. Haskell and C. T. Case, “Transient signal propagation in lossless isotropic plasmas (Report style),” USAF Cambridge Res. Lab., Cambridge, MA Rep. ARCRL-66-234 (II), 1994, vol. 2.

[24] E. E. Reber, R. L. Michell, and C. J. Carter, “Oxygen absorption in the Earth’s atmosphere,” Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (420-46)-3, Nov. 1988.

[25] (Handbook style) *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, 1985, pp. 44–60.*Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

[26] (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). *Title* (edition) [Type of medium]. Volume(issue). Available: http://www.(URL)

[27] J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com

[28] (Journal Online Sources style) K. Author. (year, month). Title. *Journal* [Type of medium]. Volume(issue), paging if given. Available: http://www.(URL)

[29] R. J. Vidmar. (1992, August). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3).* pp. 876—880. Available: http://www.halcyon.com/pub/journals/21ps03-vidmar