上海交通大学软件学院硕士论文开题报告

**基于微服务的社区治理系统**

**Micro-service Based Community Governance System**

|  |  |
| --- | --- |
| 系别 | **软件学院** |
| 学科专业 | **软件工程** |
| 研究方向 | **软件工程** |
| 姓名 | **李会一** |
| 导师 | **饶若楠** |

上海交通大学软件学院

2015年1月14日

|  |  |
| --- | --- |
| **课题来源** | 实验室横向项目 |
| **一、研究背景**   1. **研究意义**   社区治理是政府与社区组织、社区公民共同管理社区公共事务的活动。在我国, 城市社区治理模式由行政型社区向合作型社区和自治型社区的发展过程, 是社会经济体制改革和社会结构调整在城市社区发展中的一种反映,它代表着我国城市社区发展的方向。建立在合作主义基础上的新型政府与社会关系、社区制逐步取代单位制以及城市街道制体制的改革,代表着我国社区发展与制度创新的基本思路。  本文以太仓市社区治理系统为背景，该系统建立了统一的社区管理、邻里社交、便民服务平台。为社区中不同角色的居民（管理群众、生活群众、服务群众）提供了社区管理、服务以及社交的新途径。  本文以社区治理系统为研究对象，主要针对即时通讯、微服务系统架构以及统一资源管理平台进行深入研究。旨在将系统微服务化的同时利用统一资源管理系统减少微服务架构大量的管理成本，同时达到资源利用率的最大化。然而，在构建社区治理系统的过程中，遇到了如下的难题：  由于目前主流的开源资源管理平台如Apache Yarn等均不支持长时间运行的服务，因此怎样将  本文以智慧城市大数据服务平台为背景，该平台包括统一资源调度管理服务系统、运行管理服务系统、流式计算引擎、图计算引擎、内存计算引擎、Map-Reduce计算引擎，并通过上述模块的有机结合为上层应用提供服务基础，为智慧社区、智慧楼宇、智慧商圈、智慧新城、智慧政务、智慧医疗、智慧交通等智慧城市相关项目提供支持。  本论文以智慧城市统一资源调度管理服务系统为研究对象，主要为不同的分布式计算框架（引擎）进行资源调度管理机制，统一对资源进行管理，减少自行进行资源管理及调度的复杂性，并进行相应的调度算法优化，达到资源利用率最大化及最有效化。然而，在构建统一资源调度管理服务系统过程中，遇到了如下的难题：  目前的统一资源调度服务平台中灵活的调度机制较为缺乏，难以满足不同的分布式计算框架对于性能、资源占用、故障恢复等问题的需求。在这样的一个情况下，统一资源调度服务平台的灵活性、易用性难以得到体现。例如对于流式计算引擎需要将其相应节点部署至同一网络区间提高其中间节点间交互效率的问题，以及其主节点失效后重启主节点导致系统不可用的问题，目前的平台都很难给出一个较为合适的解决方案，从而导致其性能、可用性的下降。  因此，如何构建一套灵活的调度机制，适应不同的分布式计算框架，为本文的主要研究要点。   1. **研究现状分析**  Apache-Yarn Yarn(Yet Another Resource Negotiator)[2]作为hadoop[8]项目的一部分在其2.0版本时发布，旨在提高分布式集群环境下的资源利用率，这些资源包括内存、IO、网络、磁盘等。支持除了Map-Reduce[12]模型外的其他编程模型（图计算、流式计算等），提供一套普适的分布式资源调度管理从而提高整个Hadoop生态圈的广度及适应性。   * ResourceManager   ResourceManager是系统中资源的总管。其负责对系统中资源进行监控管理，用户通过ApplicationManager与ResourceManager进行协商，从而获得、管理、监控资源。同时，ResourceManager拥有可插拔式的调度器，这个调度器负责根据资源的限制，调度队列等信息来对运行的application进行调度。   * NodeManager   NodeManager是存在于每台机器上的slave进程，负责启动application的容器，监控资源使用率，并将其报告给ResourceManager。   * ApplicationManager   ApplicationMaster负责与Scheduler交互获取相应的resource容器，监控自身运行状态。  此外，Yarn还具备以下的功能：  多租户：Yarn允许多种处理引擎使用hadoop作为批处理、交互式、实时处理引擎同时运行。  集群利用：动态收集资源，减少空闲资源占用  扩展性：集群性能随着集群规模扩大线性提升  兼容性：mapreduce v1程序仍可正常在Yarn上运行 GraphX Graphx[4]是Spark[5]用于图（e.g. Web Graph and Social Network）和图并行计算（e.g. PageRank and Collaborative Filtering）的API，跟其他的分布式计算框架相比，GraphX最大的贡献是，在Spark上提供一栈式解决方案，可以方便并高效地完成图计算的一整套流水作业。  GraphX是Spark生态圈中非常重要的组件，融合了Data Parallel和Graph Parallel计算的优势，虽然在单纯的计算阶段性能不如GraphLab[9]等计算框架，但是从整个图处理流水线的视角看，就非常具有竞争性了。  目前，由于其简单易用性，及其所有中间结果均保存在Spark的RDD[6]中，GraphX已经逐渐被企业应用所亲睐。 Storm Storm[3]是一个分布式实时计算系统。可用于实时分析，在线机器学习，持续计算，分布式RPC，ETL等场景。它拥有不错的扩展性及容错能力，并保证数据一定会被处理。  Nimbus：负责资源分配和任务调度  Supervisor：负责接收Nimbus分配的任务，启动和停止属于自己管理的worker进程。  Worker：运行具体处理组件逻辑的进程。  除了上述组件之外，Storm中还有一些关键性概念：  Topology：storm中运行的一个实时应用程序，因为各个组件间的消息流动形成逻辑上的一个拓扑结构。  Spout：在一个topology中产生源数据流的组件。通常情况下spout会从外部数据源中读取数据，然后转换为topology内部的源数据。Spout是一个主动的角色，其接口中有个nextTuple()函数，storm框架会不停地调用此函数，用户只要在其中生成源数据即可。  Bolt：在一个topology中接受数据然后执行处理的组件。Bolt可以执行过滤、函数操作、合并、写数据库等任何操作。Bolt是一个被动的角色，其接口中有个execute(Tuple input)函数,在接受到消息后会调用此函数，用户可以在其中执行自己想要的操作。  Tuple：一次消息传递的基本单元。本来应该是一个key-value的map，但是由于各个组件间传递的tuple的字段名称已经事先定义好，所以tuple中只要按序填入各个value就行了，所以就是一个value list.  Stream：源源不断传递的tuple就组成了stream。 Storm on yarn Storm on yarn[16]项目由Yahoo! Inc发起，旨在将目前主流开源流式计算框架Storm迁移到Apache Yarn上运行。这样带来的好处是显而易见的：   1. 弹性计算资源：将Storm运行到Yarn上之后，Storm可与其他应用程序（例如Map-Reduce批处理应用程序）共享整个集群资源。这样，当Storm负载骤增时，可动态为它添加计算资源，而当负载减小时，可释放部分资源，从而达到资源利用率最大化。 2. 共享底层存储：Storm可与运行在Yarn框架上的其他框架共享底层的一个HDFS存储系统，可避免多个集群带来的维护成本，同时避免数据跨集群拷贝带来的网络开销和时延。 3. 支持多版本。可同时将多个版本Storm运行在YARN上，避免一个版本一个集群带来的维护成本。   在将Storm运行于Yarn上带来巨大收益的同时，当前Storm on Yarn仍然存在诸多的缺陷：   1. 难以将所有Storm服务运行在相邻的节点上，比如同一个机架上。 2. 由于Nimbus服务运行于ApplicationMaster上，一旦ApplicationMaster失败后，YARN会将其运行在另外一个节点上，这意味着Nimbus服务可能会突然在另一个节点上启动了，这给用户使用带来诸多不便。  资源调度算法 在对集群资源进行统一管理后，下一个重要的问题就是怎样对集群的资源进行有效的调度，从而达到最优的系统资源利用率。Yarn中提供的调度算法有：   1. FIFO调度算法：对任务采取先到先得的方式进行调度 2. Capacity调度算法：在共享多租户环境下使集群整个利用率达到最大 3. Fair调度算法：所有程序都获得平均的资源分配   而在越来越多的分布式计算框架部署至Yarn的需求上，这样的调度算法所能提供的灵活性已经难以满足要求。不同的计算框架拥有不同的处理模型以及不同的资源需求。  因此，更多的调度算法（Locality based scheduling, Priority-based scheduling[17], time-based scheduling, QoS based scheduling[1]等）应该不断加入Yarn当中。 | |

|  |
| --- |
| **二、研究目标**  本课题的研究目标是：以实验室承担的智慧城市项目为背景，针对智慧城市大数据平台中统一资源调度管理服务平台的多样化调度需求，结合目前主流资源管理调度框架Apache Yarn，在其基础上设计实现一套灵活的资源调度管理机制，使不同的分布式计算框架能够在该管理服务平台上获得性能、资源利用率的最优化。  围绕上述研究目标，本文的研究内容主要有：   1. 相关技术分析研究：分析并研究Spark、Storm、Graphx等相关分布式计算框架系统模型，并总结抽象其对资源调度的需求。为后续优化改进提供指导。 2. 针对Yarn资源调度管理框架进行优化，使其：    1. 可动态变更资源调度机制    2. 支持基于位置的资源调度    3. 支持基于优先级的资源调度    4. 支持独占的资源调度机制 3. 完善Storm-on-yarn项目，根据基于位置的资源调度策略，解决其目前Nimbus节点失效自动重启后，由于IP变更，导致系统不可用问题。 4. 系统测试及验证：通过系统测试及试运行应用，验证系统的可行性及有效性。 |
| **三、关键技术**  本课题研究的关键技术包括：   1. locality-based coordination[7]:   实现Yarn对Locality based调度机制的支持，修改Yarn源代码，实现节点“位置”监控功能，并开发API提供给Application以作为调度依据。   1. 调度机制优化：   实现Yarn对spark, storm, graphx的调度优化，根据其特定的资源利用机制实现不同的资源调度算法   1. 完善Storm on Yarn   在Storm on yarn项目基础上解决其目前存在的主节点失效问题，利用Yarn提供的位置信息实现locality based调度。 |
| **四、可行性分析**   1. **研究方法**   本课题的研究方法主要包括：  1. 大量阅读文献，了解各种不同的分布式计算框架的模型及业务需求，总结出最适合的调度策略  2. 研究Hadoop Yarn源码，并在其基础上对其自动调度机制进行改进  3. 研究Storm on Yarn源代码，在其基础上解决其相关问题  4. 验证性实验：对改进后的调度机制进行测试与性能评估，不管优化调度算法   1. **技术路线**   本研究将分为“基于位置的调度”，“多计算框架支持与改进”，“运行与验证”三个部分实现：   1. 基于位置的调度：   研究基于位置、基于优先级、基于时间的资源调度机制，改进Yarn使其支持该调度机制并实现。   1. 多计算框架支持与改进   针对目前市面上使用较为广泛的分布式计算框架spark，storm，graphx，使其运行于Yarn之上，并根据其特有的模型特点，实现相应的资源调度机制优化。并改进Storm on Yarn中存在的Nimbus节点失效问题，提高其可用性。   1. 运行与验证：搭建整个智慧城市统一资源管理服务系统，运行并测试上述改进的可行性及有效性。 2. **可行性分析** 3. 项目来源：由于本研究基于智慧城市大数据服务平台这一实际项目，需求数据均由甲方提供，保证了真实性和有效性。 4. 统一资源管理服务平台：目前已有比较成熟的开源资源服务管理平台Yarn，该工具可以资源管理这一基本需求，可以在其上进行优化改进。 5. 结合不同分布式计算框架：目前Yarn已支持多种不同的分布式计算框架或数据库，为本文的研究提供了基础。 6. 经费可行性：本项目成果的测试环境由实验室内部提供，不需要额外的资源，因此从经费上来说是可行的。 7. 进度可行性：整个项目有完整的安排，从调研到开发到部署，将按照开发计划有条不紊地进行，因此从进度上来说是可行的。   综上所述，本课题是可行的。 |
| **五、课题创新**  本课题的研究包括以下几个创新点：   1. 针对Yarn调度机制的改进   目前Yarn对于调度机制的支持较弱，能够提供给调度进程的有用信息仅为CPU及Memory信息，这样可实现的调度策略是极其有限的。因此，对Yarn调度机制的改进是本文的一大创新，通过向外部提供多种集群内部信息，使得多种多样的调度机制的实现成为可能。   1. 多计算框架的Yarn支持与改进：   目前Yarn虽能支持Spark、Storm等分布式计算框架的运行，但对其框架的支持并没有达到一个令人满意的程度。而这主要归结于其调度机制不够灵活。本文在实现对Yarn调度机制改进的前提下，再针对各个不同的分布式计算框架运行模型分别设计不同的资源调度方案，使其达到资源利用效率最优化。   1. Storm on Yarn开源项目改进：在上述改进的基础上，进一步解决目前Storm on Yarn项目中的顽疾主节点Nimbus节点失效问题。   本课题结合开源资源管理框架Yarn，针对其存在问题通过Storm on Yarn, Spark on Yarn等实际项目测试并进行相应的改进，是一个很有意义、很有价值的研究方向。 |
| **六、研究进度**   * 2014.1 – 2014.3查阅国内外文献，研究并学习技术，开题； * 2014.4 – 2014.5 分析Yarn内部资源调度机制，学习各种不同的资源调度算法，分析不同分布式计算框架调度需求。 * 2014.6 – 2014.7实现相应对Yarn资源调度机制的优化，开发多种资源调度算法进行测试 * 2014.8分析Storm on Yarn项目源代码，结合对Yarn的优化找出解决方案 * 2014.9 – 2014.10将解决方案应用到Storm on Yarn，并进行测试 * 2014.10 – 2014.11完成整体项目集成，进行整体评估测试 * 2014.12 撰写学位论文。 |
| **七、参考文献**  [1] Rajkumar, R.; Lee, C.; Lehoczky, J.P.; Siewiorek, D.P., "Practical solutions for QoS-based resource allocation problems," Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE , vol., no., pp.296,306, 2-4 Dec 1998  [2] Vavilapalli, Vinod Kumar, et al. "Apache hadoop yarn: Yet another resource negotiator." Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013.  [3] Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.  [4] Xin, Reynold S., et al. "Graphx: A resilient distributed graph system on spark." First International Workshop on Graph Data Management Experiences and Systems. ACM, 2013.  [5] Zaharia, Matei, et al. "Spark: cluster computing with working sets."Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. 2010.  [6] Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012.  [7] Zaharia, Matei, et al. "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling." Proceedings of the 5th European conference on Computer systems. ACM, 2010.  [8] Shvachko, Konstantin, et al. "The hadoop distributed file system." Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010.  [9] Low, Yucheng, et al. "Graphlab: A new framework for parallel machine learning." arXiv preprint arXiv:1006.4990 (2010).  [10] [Hitesh Ballani , Paolo Costa , Thomas Karagiannis , Ant Rowstron, Towards predictable datacenter networks, ACM SIGCOMM Computer Communication Review, v.41 n.4, August 2011](http://dl.acm.org/citation.cfm?id=2018465&CFID=617641976&CFTOKEN=19180186)  [11] [Ronnie Chaiken , Bob Jenkins , Per-Åke Larson , Bill Ramsey , Darren Shakib , Simon Weaver , Jingren Zhou, SCOPE: easy and efficient parallel processing of massive data sets, Proceedings of the VLDB Endowment, v.1 n.2, August 2008](http://dl.acm.org/citation.cfm?id=1454166&CFID=617641976&CFTOKEN=19180186)  [12][Jeffrey Dean , Sanjay Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM, v.51 n.1, January 2008](http://dl.acm.org/citation.cfm?id=1327492&CFID=617641976&CFTOKEN=19180186)  [13] [Michael Isard , Mihai Budiu , Yuan Yu , Andrew Birrell , Dennis Fetterly, Dryad: distributed data-parallel programs from sequential building blocks, Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, March 21-23, 2007, Lisbon, Portugal](http://dl.acm.org/citation.cfm?id=1273005&CFID=617641976&CFTOKEN=19180186)  [14] S. Ghemawat, H. Gobioff, S. Leung. "The Google file system," In Proc. of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp 29-43.  [15] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. Technical Report UCB/EECS-2010-87, EECS Department, University of California, Berkeley, May 2010.  [16] The Yahoo! Blog: https://developer.yahoo.com/blogs/ydn/storm-yarn-released-open-source-143745133.html  [17] Song, Ying, et al. "A service-oriented priority-based resource scheduling scheme for virtualized utility computing." High Performance Computing-HiPC 2008. Springer Berlin Heidelberg, 2008. 220-231.  [18] Sha, Lui, et al. "Real time scheduling theory: A historical perspective." Real-time systems 28.2-3 (2004): 101-155.  [19] Hunt, Patrick, et al. "ZooKeeper: Wait-free Coordination for Internet-scale Systems." USENIX Annual Technical Conference. Vol. 8. 2010.  [20] S4 Distributed stream computing platform. http://incubator.apache.org/s4/  [21] Spark Streaming. http://spark.incubator.apache.org/docs/latest/streaming-programming-guide.html  [22] Gonzalez, Joseph E., et al. "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs." OSDI. Vol. 12. No. 1. 2012.  [23] Facebook Engineering. Under the Hood: Scheduling MapReduce jobs more efficiently with Corona, November 2012.  [24] Schwarzkopf, Malte, et al. "Omega: flexible, scalable schedulers for large compute clusters." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.  [25] Xu, Yiqi, Adrian Suarez, and Ming Zhao. "Ibis: interposed big-data i/o scheduler." Proceedings of the 22nd international symposium on High-performance parallel and distributed computing. ACM, 2013.  [26] Wang, Ke, and Ioan Raicu. "Scheduling Data-intensive Many-task Computing Applications in the Cloud."  [27] Narayanamurthy, Shravan, et al. "Towards resource-elastic machine learning." (2013).  [28] Schuh, Hannes. "Collecting Execution Statistics of Scientific Workflow on Hadoop YARN."Dowling, Jim. "Attack of the Borg: Decentralized Resource Management." (2014).  [29] Ivanov, Todor, Nikolaos Korfiatis, and Roberto V. Zicari. "On the inequality of the 3V's of Big Data Architectural Paradigms: A case for heterogeneity." arXiv preprint arXiv:1311.0805 (2013).  [30] Raj, R. Sandeep, and G. Prabhakar Raju. "An Enhanced Approach for Resource Management Optimization in Hadoop."  [31] Kromonov, Ilja, Pelle Jakovits, and Satish Narayana Srirama. "NEWT-A resilient BSP framework for Iterative algorithms on hadoop YARN." High Performance Computing & Simulation (HPCS), 2014 International Conference on. IEEE, 2014. |