NanoLog: A Nanosecond Scale Logging System

专业：计算机技术硕1812 姓名：武新楠 学号：M201877271

# 背景

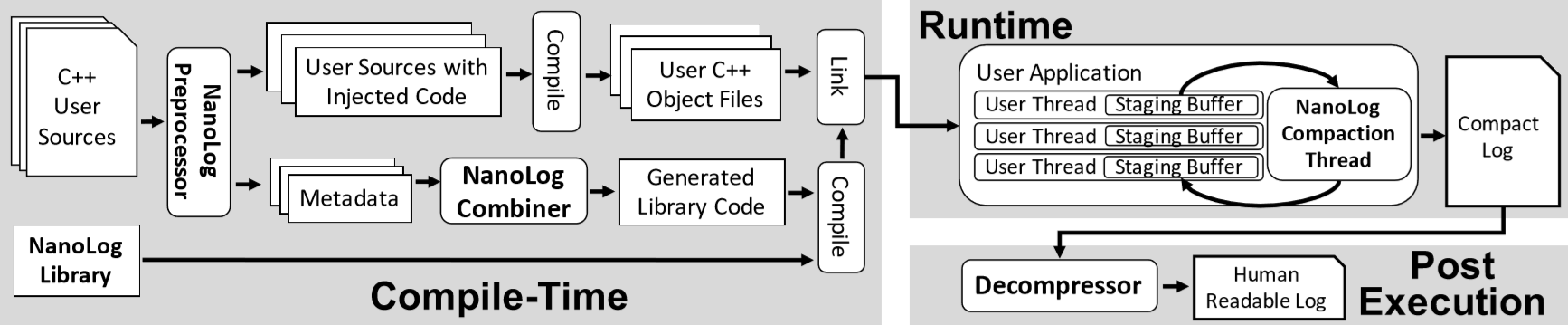
日志记录系统允许开发人员在执行期间生成人类可读的应用程序追踪。开发人员使用日志记录语句来注释系统代码。每个日志记录语句都使用类型printf的接口来指定一个静态字符串，指示刚刚发生的事情以及与事件相关的一些运行时数据。然后，日志记录系统添加补充信息，例如事件发生的时间、源代码文件和日志记录语句的行号、严重性级别以及日志记录线程的标识符。

最简单的日志记录实现是同步输出每个日志消息，与应用程序的执行一致。由于两个原因，这种方法的性能相对较低。首先，格式化日志消息通常需要0.5-1μs（1000-2000个周期）。在低延迟服务器中，这可能代表一个重要的请求的总服务时间的一小部分。其次，I/O开销很大。日志消息通常为50-100字节长，因此具有250MB/S带宽的闪存驱动器每秒只能吸收几百万条消息。此外，应用程序有时必须进行内核调用以刷新日志缓冲区，这将引入额外的延迟。

NanoLog的设计源自两个关于日志的观察。第一，完全格式化的人类可读消息不一定需要在应用程序内部生成。相反，应用程序可以记录每条消息的原始组件，当人们需要时，可以在以后生成人类可读的消息。许多日志永远不会被人类读取，在这种情况下，可以跳过消息格式化步骤。读取日志时，通常只检查一小部分消息，例如崩溃时的消息，因此只需要对一小部分日志进行格式化。最后，许多日志由分析引擎处理。在这种情况下，处理原始数据要比人类可读的日志版本快得多。

# 2. 原理

Nano的低延迟来自于在编译时执行工作以从日志消息中提取静态组件并将格式化延迟到离线过程。因此，Nano系统分解为三个组件，如图1所示：



图一

预处理器：在编译时从日志消息中提取和编目静态组件，用优化的代码替换原始日志记录语句，为每个日志消息生成唯一的压缩函数，并生成输出静态信息字典的函数。

运行时库：为来自多个日志记录线程的日志消息的基础结构提供缓冲，并使用生成的压缩和字典函数以紧凑的二进制格式输出日志。

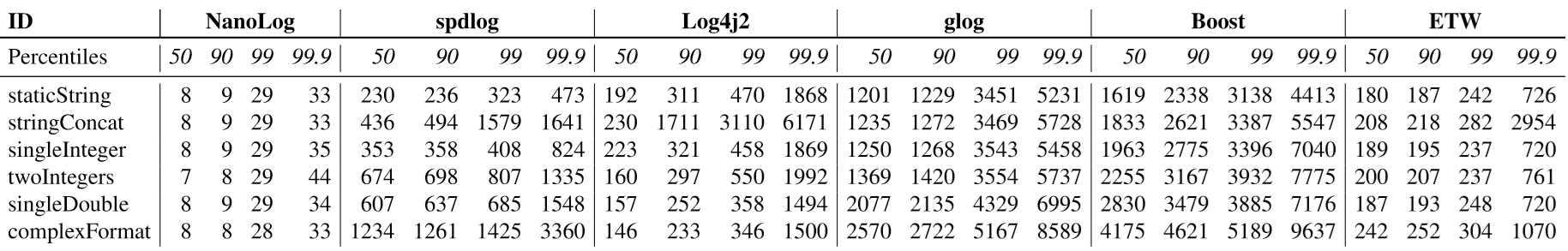
解压缩器：将紧凑的二进制文件与字典中的静态信息重新组合，以将日志扩展为人类可读的格式，或者对日志内容进行分析。

# 3. 性能

我们选择将NanoLog与Log4j2，spdlog，glog，Boost日志和Windows事件追踪（ETW）进行对比。为了确保我们选择的日志消息代表真实世界的使用，我们静态分析了来自五个开源系统的日志语句。得出日志消息平均具有大约45给字符的静态内容，并且整数是最常见的动态类型。字符串是最常见的类型，但仔细观察后，大多数使用的字符串都可以从NanoLog的静态提取方法中受益。

## 3.1 吞吐量

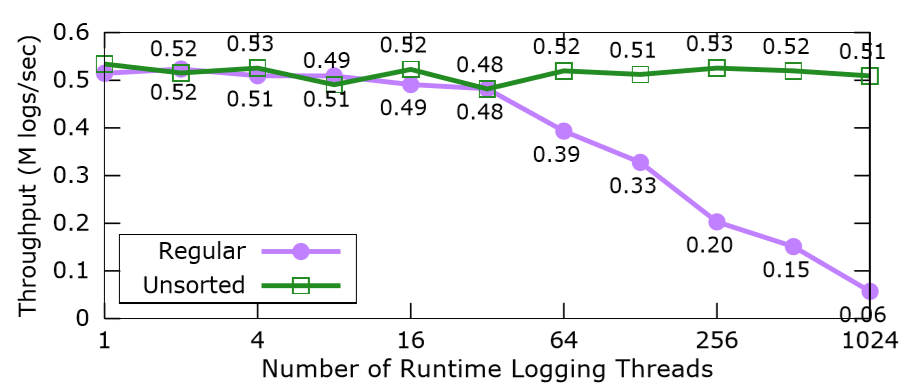
图二显示了NanoLog，spdlog，Log4j2，Boost，glog和ETW实现的最大吞吐量。NanoLog比其他系统快1.8-133倍。



图二

## 3.2 解压

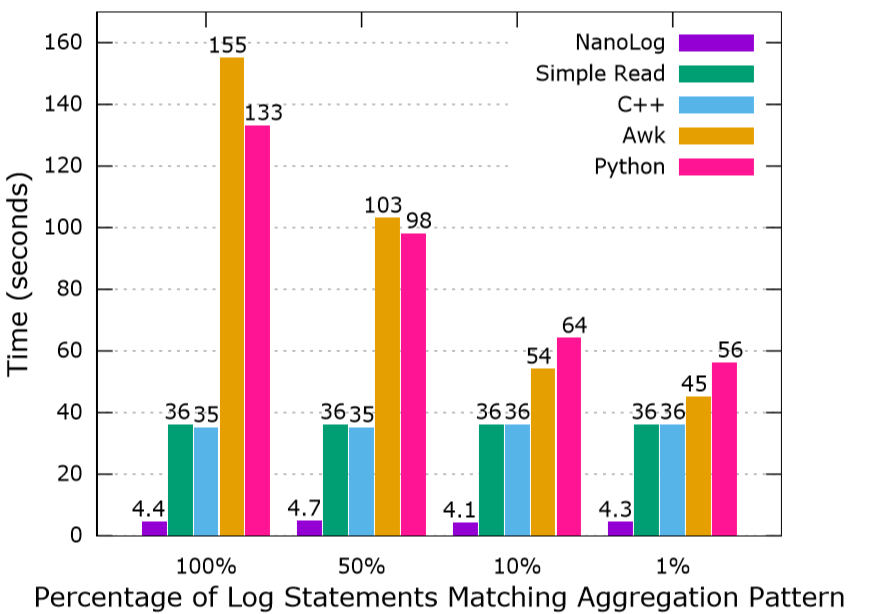
解压缩器当前使用简单的单线程实现，它可以在大约0.5M日志消息/秒的峰值处解压缩。传统系统（如Log4j2）可以在运行时实现超过2M日志消息/秒的更高吞吐量，因为他们利用所有日志记录线程进行格式化。NanoLog的解压缩程序可以修改为使用多个线程来实现更高的吞吐率。如图三。



图三

## 3.3 聚合性能

NanoLog的紧凑型二进制日志输出承诺比完整的未压缩对应物更有效地进行日志聚合/分析。图四显示了此聚合超过100M日志消息的执行时间。NanoLog比其他系统快了进一个数量级，平均缩短4.4秒来压缩紧凑的日志文件，而其他系统则需要35秒。



图四

# 4. 限制

NanoLog的一个限制是它目前只能在静态类似printf的格式字符串上运行。这意味着动态格式字符串，c++流和toString（）方法不会受益于NanoLog。虽然我们没有动态格式字符串的高性能解决方案，但我们相信更强大的预处理器/编辑器扩展可能能够通过查看类型或提供类似snprintf来从c++流中提取模式用于为NanoLog生成中间件。

最后，NanoLog目前假设日志存储在本地文件系统中。但是，可以轻松修改它以远程存储日志。在这种情况下，NanoLog的吞吐量将受到网络吞吐量和远程存储机制的限制。大多数结构化存储系统（如数据库）都足够慢，严重限制了NanoLog的性能。

# 5.参考文献

[1] BOLOSKY, W. J., AND SCOTT, M. L. False Sharing and Its Effect on Shared Memory Performance. In USENIX Systems on USENIX Experiences with DistributedandMultiprocessorSystems-Volume4 (Berkeley, CA, USA, 1993), Sedms’93, USENIX Association, pp. 3–3.

[2] boost C++ libraries. <http://www.boost.org>.

[3] Datadog. <https://www.datadoghq.com>.

[4] DRAGOJEVI´ C, A., NARAYANAN, D., CASTRO, M., AND HODSON, O. FaRM: Fast Remote Memory. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) (Seattle, WA, Apr. 2014), USENIX Association, pp. 401–414.

[5] ECMASCRIPT, ECMA AND EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION AND OTHERS. Ecmascript language speciﬁcation, 2011.

[6] ERLINGSSON, ´U., PEINADO, M., PETER, S., BUDIU, M., AND MAINAR-RUIZ, G. Fay: extensibledistributedtracingfromkernelstoclusters. ACM Transactions on Computer Systems (TOCS) 30, 4 (2012), 13.

[7] FELDERMAN, B. Personal communication, June 2015. Google.

[8] FONSECA, R., PORTER, G., KATZ, R. H., SHENKER, S., AND STOICA, I. X-trace: A pervasive network tracing framework. In Proceedings of the 4th USENIX conference on Networked systems design&implementation(2007),USENIXAssociation, pp. 20–20.

[9] GAILLY, J.-L., AND ADLER, M. gzip. http: //www.gzip.org.

[10] GNU COMMUNITY. Using the GNU Compiler Collection: Declaring Attributes of Functions. https://gcc.gnu.org/onlinedocs/gcc3.2/gcc/Function-Attributes.html, 2002.

[11] GOOGLE. glog: Google Logging Module. <https://github.com/google/glog>.

[12] GOOGLE. gRPC: A high performance, opensource universal RPC framework. http:// [www.grpc.io](http://www.grpc.io).

[13] GREGG, B. Linux bpf superpowers. http: //www.brendangregg.com/blog/201603-05/linux-bpf-superpowers.html, 2016.

[14] GREGG, B., AND MAURO, J. DTrace: Dynamic TracinginOracleSolaris,MacOSXandFreeBSD. Prentice Hall Professional, 2011.

[15] HOLLINGSWORTH, J. K., MILLER, B. P., AND CARGILLE, J. Dynamic program instrumentation for scalable performance tools. In Scalable HighPerformance Computing Conference, 1994., Proceedings of the (1994), IEEE, pp. 841–850.

[16] INTEL, R. Intel 64 and ia-32 architectures optimizationreferencemanual. IntelCorporation,June (2016).

[17] JTC, I. SC22/WG14. ISO/IEC 9899: 2011. InformationTechnology Programminglanguages C. (2011).

[18] LIM, H., HAN, D., ANDERSEN, D. G., AND KAMINSKY, M. Mica: A holistic approach to fast in-memory key-value storage. USENIX.

[19] The Linux Kernel Organization. https: //www.kernel.org/nonprofit.html, May 2018.

[20] LMAX Disruptor: High Performance InterThread Messaging Library. http://lmaxexchange.github.io/disruptor/.

[21] MACE, J., ROELKE, R., AND FONSECA, R. Pivot tracing: Dynamic causal monitoring for distributed systems. In Proceedings of the 25th Symposium on Operating Systems Principles (New York, NY, USA, 2015), SOSP ’15, ACM, pp. 378–393.

[22] memcached: a Distributed Memory Object Caching System. http:// www.memcached.org/, Jan. 2011.

[23] MICROSOFT. Wpp software tracing. https://docs.microsoft.com/en-us/ windows-hardware/drivers/devtest/ wpp-software-tracing, 2007.

[24] MORTORAY, E. Wait-free queueing and ultra-low latency logging. https://mortoray.com/ 2014/05/29/wait-free-queueingand-ultra-low-latency-logging/, 2014.

[25] NAGARAJ, K., KILLIAN, C., AND NEVILLE, J. Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (2012), USENIX Association, pp. 26–26.

348 2018 USENIX Annual Technical Conference USENIX Association

[26] NORVIG, P. Natural Language Corpus Data: Beautiful Data, 2011 (accessed January 3, 2018).

[27] OLINER, A., GANAPATHI, A., AND XU, W. Advances and challenges in log analysis. Communications of the ACM 55, 2 (2012), 55–61.

[28] OTT, D. Personal communication, June 2015. VMWare.

[29] OUSTERHOUT, J., GOPALAN, A., GUPTA, A., KEJRIWAL, A., LEE, C., MONTAZERI, B., ONGARO, D., PARK, S. J., QIN, H., ROSENBLUM, M., ET AL. TheRAMCloudStorageSystem. ACM Transactions on Computer Systems (TOCS) 33, 3 (2015), 7.

[30] PAOLONI, G. How to benchmark code execution times on intel ia-32 and ia-64 instruction set architectures. Intel Corporation, September 123 (2010).

[31] PARK, I., AND BUCH, R. ImproveDebuggingAnd PerformanceTuningWithETW. MSDNMagazine, April 2007 (2007).

[32] Performance Utilities. https://github.com/ PlatformLab/PerfUtils.

[33] printf - C++ Reference. http:// www.cplusplus.com/reference/cstdio/ printf/.

[34] Redis. <http://redis.io>.

[35] SIGELMAN, B. H., BARROSO, L. A., BURROWS, M., STEPHENSON, P., PLAKAL, M., BEAVER, D., JASPAN, S., AND SHANBHAG, C. Dapper, a large-scale distributed systems tracing infrastructure. Tech. rep., Technical report, Google, 2010.

[36] SLEE, M., AGARWAL, A., AND KWIATKOWSKI, M. Thrift: Scalablecross-languageservicesimplementation. Facebook White Paper 5, 8 (2007).

[37] Snappy, a fast compressor/decompressor. https: //github.com/google/snappy.

[38] spdlog: ASuperfastC++logginglibrary. https: //github.com/gabime/spdlog.

[39] Splunk. <https://www.splunk.com>.

[40] STALLMAN, R. M., MCGRATH, R., AND SMITH, P. GNU Make: A Program for Directed Compilation. Free software foundation, 2002.

[41] THE APACHE SOFTWARE FOUNDATION. Apache HTrace: A tracing framework for use with distributed systems. http://htrace.incubator.apache.org.

[42] THE APACHE SOFTWARE FOUNDATION. Apache HTTP Server Project. <http://httpd.apache.org>.

[43] THE APACHE SOFTWARE FOUNDATION. Apache Log4j 2. https://logging.apache.org/ log4j/log4j-2.3/manual/async.html.

[44] THE APACHE SOFTWARE FOUNDATION. Apache Spark. <https://spark.apache.org>.

[45] THE APACHE SOFTWARE FOUNDATION. Log4j2 Location Information. https://logging.apache.org/ log4j/2.x/manual/ layouts.html#LocationInformation.

[46] VARDA, K. Protocol buffers: Googles data interchange format. Google Open Source Blog, Available at least as early as Jul (2008).

[47] YANG, S. NanoLog: an extremely performant nanosecond scale logging system for C++ that exposes a simple printf-like API. https:// github.com/PlatformLab/NanoLog.

[48] Twitter Zipkin. <http://zipkin.io>.

[49] ZIV, J., AND LEMPEL, A. A universal algorithm for sequential data compression. IEEE Transactions on information theory 23, 3 (1977), 337–343.