

Reinventing Grep with Parabix Technology

January 18, 2020

1 Introduction

Searching through text files for strings matching a particular pattern is a common task in information processing. The software tool `grep` was developed by Ken Thompson for this purpose and made available with Unix Version 4 in 1973. [McI87]. The name `grep` arises from the `g/re/p` command of Thompson's `ed` editor meaning globally search for regular expression and print. While Thompson's initial version did not support the regular expression features of alternation and grouping, this was remedied by the development of `egrep` by Alfred Aho in 1975 [Hum88]. Andrew Hume improved `egrep` performance by incorporating the Boyer-Moore algorithm for fixed strings in 1988. Free and open source software versions of `grep` were developed by the GNU and BSD projects. Bitwise parallel NFA simulation was introduced with `agrep` [WM92] and `nrgrep` [Nav01].

All of these `grep` implementations are fundamentally sequential in nature, primarily building on automata theory techniques involving deterministic or nondeterministic finite automata (NFAs or DFAs). In general, the searches proceed byte-at-a-time updating the automaton state (or set of states in the case of NFAs) with each step. The byte-at-a-time processing model was well matched to processor capabilities at the time of initial `grep` development, but processors have evolved to be able to process many bytes in parallel using short vector SIMD instructions. For more than two decades, processing 16 bytes at a time has been possible with the Intel SSE instructions, Power PC AltiVec instructions or ARM Neon instructions. In the last several years, Intel has further increased SIMD register width, first with 32-byte AVX2 technology and most recently with 64-byte AVX-512 technology.

Parabix technology is a programming framework under development by our research group at Simon Fraser University to take advantage of the wide SIMD registers for streaming text processing applications such as `grep` [Lin+12]. Rather than focussing on byte-oriented processing, however, it uses the concept of bitwise data parallelism. In this model, bit positions within SIMD registers are associated with byte positions in input data streams. With AVX2 technology, for example, the goal is to process 256 bytes at a time with the 256-bit SIMD registers. We have applied Parabix technology to accelerate Unicode

transcoding [Cam08], XML parsing [Cam+11; Med+13] and regular expression search [Cam+14], while other groups have applied related techniques to accelerate RNA protein search [PMG10] and JSON parsing [Li+17].

As a showcase of the Parabix framework, icgrep is a new grep implementation fundamentally built using bitwise data parallel techniques. With our promising performance results in icgrep 1.0 [Cam+14], we have focused on building a fully modern grep with broad support of extended regular expression features including the lookaround assertions of Perl-compatible regular expressions and the full set of Unicode level 2 regular expression features defined by the Unicode consortium [DH16]. We believe icgrep is the first grep implementation to achieve this level of Unicode support. In addition, we have also focused on incorporating educational features so that users can explore and display various transformations that take place during regular expression processing and/or Parabix compilation.

From a performance perspective, we have considered three important aspects for modern software tools. One aspect is the use of fundamentally parallel algorithms as well as the systematic use of SIMD and multicore parallelism. The second aspect is to focus on scalability, that is to arrange to automatically take advantage of both available SIMD register width and available cores to achieve performance that scales with these resources. While there is still considerable work to do in the development of our framework, icgrep does indeed demonstrate such scalability, even for single file search. From this perspective, we think that icgrep represents an interesting initial data point with respect to software tool scalability. Finally, we also have focused on consistent and predictable performance, particularly in response to the rise in denial-of-service attacks that exploit pathological cases for many existing regular expression tools [KRT13].

While research continues, our goal is to introduce icgrep as an important practical contribution that pushes the boundaries on both the performance and capability of grep search software. As with all open source software projects, we expect to refine the software over time and invite collaborators to join our effort.

2 Parabix Regular Expression Matching

References

- [McI87] MD McIlroy. “A Research Unix reader: annotated excerpts from the Programmer’s Manual, 1971–1986 (PDF)(Technical report)”. In: *CSTR. Bell Labs* 139 (1987).
- [Hum88] Andrew Hume. “A tale of two greps”. In: *Software: Practice and Experience* 18.11 (1988), pp. 1063–1072.
- [WM92] Sun Wu and Udi Manber. “Agrep—a fast approximate pattern-matching tool”. In: *Usenix Winter 1992 Technical Conference*. 1992, pp. 153–162.

- [Nav01] Gonzalo Navarro. “NR-grep: a fast and flexible pattern-matching tool”. In: *Software: Practice and Experience* 31.13 (2001), pp. 1265–1312.
- [Cam08] Robert D Cameron. “A case study in SIMD text processing with parallel bit streams: UTF-8 to UTF-16 transcoding”. In: *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*. ACM. 2008, pp. 91–98.
- [PMG10] Robert J Peace, H Mahmoud, and James Robert Green. “Exact string matching for MS/MS protein identification using the Cell Broadband Engine”. In: *CMBES Proceedings* 33 (2010).
- [Cam+11] Robert D Cameron et al. “Parallel scanning with bitstream addition: An xml case study”. In: *European Conference on Parallel Processing*. Springer. 2011, pp. 2–13.
- [Lin+12] Dan Lin et al. “Parabix: Boosting the efficiency of text processing on commodity processors”. In: *IEEE International Symposium on High-Performance Comp Architecture*. IEEE. 2012, pp. 1–12.
- [KRT13] James Kirrage, Asiri Rathnayake, and Hayo Thielecke. “Static analysis for regular expression denial-of-service attacks”. In: *International Conference on Network and System Security*. Springer. 2013, pp. 135–148.
- [Med+13] Nigel Medforth et al. “icXML: Accelerating a commercial XML parser using SIMD and multicore technologies”. In: *Balisage: The Markup Conference*. 2013, pp. 6–9.
- [Cam+14] Robert D Cameron et al. “Bitwise data parallelism in regular expression matching”. In: *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*. IEEE. 2014, pp. 139–150.
- [DH16] Mark Davis and Andy Heninger. “Unicode regular expressions”. In: *Unicode Technical Recommendation* 18 (2016).
- [Li+17] Yinan Li et al. “Mison: a fast JSON parser for data analytics”. In: *Proceedings of the VLDB Endowment* 10.10 (2017), pp. 1118–1129.