Ashtadhyayi—An Experimental Approach to Enhance Programming Languages and Compiler Design Using Panini's Grammar

A. Soumya Mahalakshmi and Minal Moharir

Abstract The astonishing foresight and vision of ancient scientists remain unparalleled. They augmented the conversations and thoughts about philosophy and science, often revealing structured explanations for phenomena that were way ahead of their time. One such instance is the emergence of the Sanskrit language which was characteristic of the literature during the Vedic period in Ancient India, in third century BC. Sanskrit has been widely accepted as an extremely logical language and the sole credit for this goes to Panini, Sanskrit Grammarian and vastly regarded as the first programmer of the world. In his work *Ashtadhyayi*, he summarizes the logic and grammar for Sanskrit in the form of 4000 *Sutras*, effectively building a machine that generates thousands of Sanskrit words and sentences. While linguists around the world have begun realizing similarities to Backus-Naur form in *Ashtadhyayi*, Sanskrit is being claimed as the best language for Artificial Intelligence and Natural Language Processing. The aim of this project is to identify and understand the *Sutra* style of Panini Grammar, and to exploit the optimizations in language for better programming languages. The effectiveness of applying a similar optimized grammar for use in C programming language has been explored. Further, the results have been extrapolated to understand the advantage of using this grammar in CUDA-C in a graphical processing unit. The performed experiments validate the efficiency and pronounced the enhancement of using a Panini-inspired compiler for C as well as CUDA-C, which can lead to path-breaking speed and a new paradigm to approach fast data technologies using next generation GPU systems.

Keywords Panini · Ashtadhyayi · Programming languages
C · CUDA-C · Compiler design · Grammar · Sanskrit · Sutras

A. Soumya Mahalakshmi (✉) · M. Moharir
Department of Computer Science and Engineering, R V College of Engineering,
Bengaluru, Karnataka, India
e-mail: a.soumyamahalakshmi@gmail.com

P. K. Sa et al. (eds.), *Recent Findings in Intelligent Computing Techniques*,
Advances in Intelligent Systems and Computing 709,
https://doi.org/10.1007/978-981-10-8633-5_1

1 Introduction

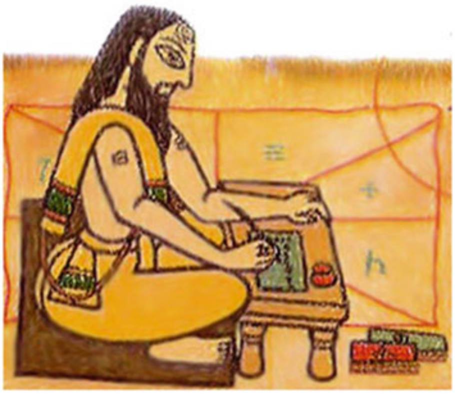
The Indian Civilization was an ardent contributor to the ancient understanding of science and technology. The astonishing foresight and vision of ancient scientists remain unparalleled. They augmented the conversations and thoughts about philosophy and science, often revealing structured explanations for phenomena that were way ahead of their time. Ancient India was a land of scholars, sages, seers, and scientists. Research has revealed outstanding examples of genius inventions such as the mathematical digit "zero". Aryabhata created a symbol for zero and its integration into the place-value system enabled one to write large numbers using only ten symbols. Binary numbers involve the most rudimentary understanding of computer science, and it was first described by the Pingala in Chandahśāstra, a Sanskrit treatise on prosody. Another instance in relation to computer science is the chakravala method, which evolved as a cyclic algorithm to obtain integer solutions for intermediate quadratic equations. Research has suggested the presence of explanations and origins of theories such as the atomic theory and heliocentric theory in Ancient Indian texts long before the rest of the world documented it. A similar observation can be made regarding the knowledge of metallurgy, surgery, rocket science, and medicine.

In this regard, it is worth mentioning the achievement of Panini, who has gained the reputation of being the pioneer in grammars and languages. He was hailed as a Sanskrit grammarian who worked on phonetics, phonology, and morphology, thereby providing a complete and comprehensive grammar for Sanskrit. Incidentally, the word "Sanskrit" means "complete" or "perfect" and it was thought of as the divine language, or language of the gods. The Sanskrit language was indicative of the Vedic period in India, and was used in nearly all scientific and literary documents of the time. In several contexts, Panini has been identified as the world's first programmer, owing to his outstanding contribution in defining a structured grammar for languages. Today, linguists and programmers around the world have identified the merits of his grammar used in the Sanskrit language, which in many ways helps present day understanding of Natural Language Processing and Artificial Intelligence (Fig. 1).

Panini's work is largely compiled in his treatise called *Astadhyayi*, whose name indicates the presence of eight chapters, each subdivided into quarter chapters. Panini gives formal production rules and definitions to describe Sanskrit grammar. Panini's constructions of the Sanskrit language using a grammar were vastly similar to a mathematical function, which integrated 1700 basic elements such as nouns, verbs, etc. [1].

There has been considerable speculation about the possibility of the presence of rules in ancient Indian logic and grammar that can aid advancement in cognitive and computer science. The significance of the context-sensitive grammars of Panini was understood only when the Chomsky normal form was introduced in the nineteenth century. Formally, Panini's grammar, Ashtadhyayi is studied together with the dhatupatha (a list of verbal roots arranged into sublists), and the ganapatha (a list of various classes of morphs). Ashtadhyayi provides a structure for the

Fig. 1 An artist's interpretation of Panini



Courtesy : India's Postal Stamp of Panini, 2004

analysis of sentences, which has been described as a machine generating words and sentences of Sanskrit [2].

Sanskrit's phonology, morphology, and syntax are described in Ashtadhayayi in a collection of 4000 sutras, which define its structure as a rule-based system. Each sutra is a reference to a rule, which can be definitions, theorems (linguistic facts), and meta-theorems (rules regarding rules). Since Ancient India followed a mechanism of oral propagation of tradition, it was important to make the 4000 sutras as concise as possible, consisting of only three words each, achieved through various optimizations. Hence, it is generally agreed that the Paninian system is based on a principle of economy, an Occam's razor. This makes the structure to be of special interest to cognitive scientists [2]. Traditionally, a sutra is defined as the most concise of statements which uses as few letters as possible. The Grammar follows a principle of the following form:

iko yan aci

Each sutra can be analogous to a production rule that defines a grammar. Therefore, it is only fitting to observe that the Panini system advocates the use of only three tokens in a production rule, achieved through employing several algebraic devices such as prefixes and suffixes. Panini took the idea of providing a context for action in terms of its relations to agents and situation, given by the karaka theory. The following semantic notions capture the various aspects of action, which have been used for optimizations [3]:

a. Apadana: that which is fixed when departure takes place
b. Sampradana: the recipient of the object
c. Karana: the main cause of the effect; instrument
d. Adhikarana: the basis, location
e. Karman: what the agent seeks most to attain; deed, object
f. Kartr: one who is independent; the agent

Here, any sentence is optimized to only three tokens, using prefixes, suffixes, recursion, and context. However, in a regular C compiler, there is no upper limit on the number of tokens in a statement. Therefore, in this study, an attempt has been made to employ these ideas to reduce the number of tokens in a production rule that defines a grammar for programming languages. Hence, LEX and YACC tools were used to define a limited functionality compiler for C programming language, which was designed using grammar production rules influenced by these optimizations.

2 Methodology

A. *Understanding*

1. To analyze the sutra style of grammar which is highly optimized using prefixes and suffixes
2. To ponder over the present grammar of programming languages such as C and CUDA-C

B. *Design*

1. To develop a C compiler using YACC for the new Panini inspired grammar
2. To develop a matrix multiplication program in C, with the new compiler

C. *Validation*

1. To compile and run the program against a conventional compiler and the Panini compiler
2. To compare, prorate, and analyze the results obtained (Fig. 2).

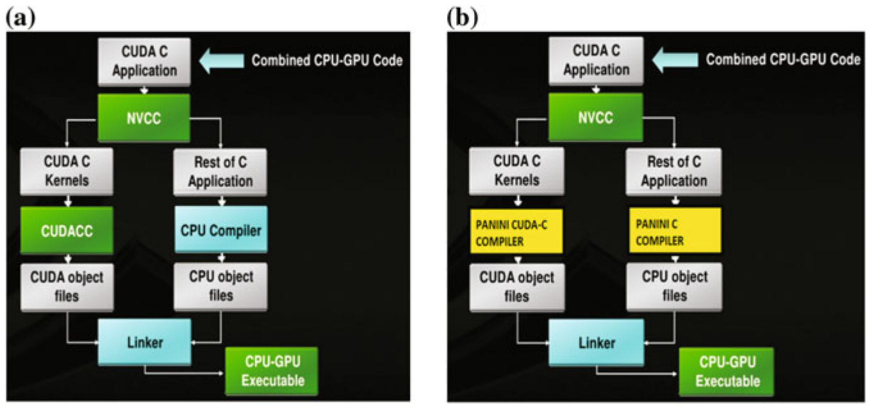


Fig. 2 Design of compiler design of **a** conventional CUDA-C compiler **b** optimized Panini CUDA-C compiler

Table 1 Experimental results for optimization using Panini compiler

CUDA-C language (s)	C language (s)	Type of compiler
1.01	44.1537	Conventional compiler
0.6059 (Prorated)	26.4922	Optimized compiler

3 Experimental Results

A C program for matrix multiplication was compiled and executed over a conventional C compiler, and then with a compiler optimized using Panini grammar and time taken was 44.1537 s and 26.4922 s, respectively, for input size 2000. When executed on NVIDIA Tesla K40, the time taken to execute matrix multiplication on a CUDA compiler was 1.01 s. Through proration, we can reduce the time taken to 0.6059 s (Table 1).

4 Conclusion

i. It can be concluded that using Panini's optimizations in the grammar of a compiler can bring about a marked speed enhancement of nearly 40–50%.
ii. When implemented for CUDA compilers, it can revolutionize fast data technologies for next generation GPU systems.
iii. Panini has indeed proved himself to be the world's first programmer.

Acknowledgements We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU as a part of GPU Research Center, used for this research.

References

1. O'Connor, J.J., Robertson, E.F.: "Panini", School of Mathematics and Statistics, University of St Andrews, Scotland
2. Bhate, S., Kak, S.: Panini's grammar and computer science. Ann. Bhandarkar Orient. Res. Inst. **72**, 79–94 (1993)
3. Kak, Subhash C.: The Paninian approach to natural language processing. Int. J. Approximate Reasoning **1**(1), 117–130 (1987)