Annual Report for Period: 09/2009 - 08/2010
Submitted on: 06/17/2010
Principal Investigator: Yang, Jun .
Award ID: 0916027
Organization: Duke University
Submitted By:
Yang, Jun - Principal Investigator
Title:
III: Small: RIOT: Statistical Computing with Efficient, Transparent I/O

Project Participants

Senior Personnel

Name: Yang, Jun
Worked for more than 160 Hours: Yes
Contribution to Project:
Jun Yang has been serving as the faculty lead on this project.

Post-doc

Graduate Student

Name: Zhang, Yi
Worked for more than 160 Hours: Yes
Contribution to Project:
Since the project's conception Yi Zhang has been the main student contributor. He has been involved in most of the research and development efforts, including the RIOT-DB system (CIDR 2009) and the next generation RIOT system (ICDE 2010). Yi is currently focusing on optimizing array storage and processing in RIOT.

Name: Thonangi, Risi
Worked for more than 160 Hours: Yes
Contribution to Project:
Risi Thonangi has been researching how to leverage the recently emerging SSDs (solid state drives) to improve performance. He is currently developing an index for SSDs that support efficient concurrent accesses despite the need for batch index organization.

Name: Herodotou, Herodotos
Worked for more than 160 Hours: Yes
Contribution to Project:
Although not funded by this project, Herodotos Herodotou contributed to the development of the RIOT-DB system (CIDR 2009).

Undergraduate Student

Name: Zhang, Weiping
Worked for more than 160 Hours: Yes
Contribution to Project:
Shortly after the conception of the project, Weiping Zhang joined the RIOT team and worked for an REU-funded research internship and an independent study course (Spring 2010). He contributed to the development of RIOT's native array storage manager and is a co-author on of ICDE 2010 demo paper.

Name: Yan, Jiaqi

Worked for more than 160 Hours: No

Contribution to Project:

Jiaqi Yan joined the RIOT team in the summer of 2010 on an REU-funded research internship. He is studying several information retrieval applications with scalability demands to see how RIOT might help.

Technician, Programmer

Other Participant

Research Experience for Undergraduates

Organizational Partners

HP Labs

Other Collaborators or Contacts

Min Wang (HP Labs China), Meichun Hsu (HP Labs Palo Alto)

Activities and Findings

Research and Education Activities:

Recent technological advances have enabled collection of massive amounts of data in science, commerce, and society. These large datasets have brought us closer than ever before to solving important problems such as decoding human genomes and coping with climate changes. Meanwhile, the exponential growth in the amount of data has created an urgent challenge. Today, much of advanced analysis is done with programs custom-developed by statisticians. Unfortunately, progress has been hindered by the lack of easy-to-use statistical computing environments that scale to large datasets. Many existing tools assume that datasets fit in main memory; when applied to large datasets, they are unacceptably slow because of excessive disk input/output (I/O) operations. There have been many approaches toward I/O-efficiency, but none has gained traction with the statistical computing community. Disk-based storage engines and I/O-efficient function libraries provide only a partial solution, because many sources of I/O-inefficiency in a program remain at a higher, inter-operation level: e.g., how large intermediate results are passed between operations, how much performance can be gained by deferring and reordering operations, etc. Database systems seem to be a natural solution, with I/O-efficiency and a high-level language (SQL) enabling many high-level optimizations. However, work in integrating databases and statistical computing has mostly remained database-centric, forcing statisticians to learn unfamiliar languages and deal with their impedance mismatch with the host language.

To make a practical impact on the statistical computing community, the RIOT project seeks to extend R---an open-source statistical computing

environment widely used by statisticians---to transparently provide scalability over large datasets. Transparency means no SQL, or any new language to learn. Transparency means that existing code should run without modification, and automatically gain efficiency. RIOT is developing an end-to-end solution that addresses issues on all fronts: I/O-efficient and parallel algorithms, deferred evaluation, pipelined execution, cost-driven optimization, smart storage and materialization options, and seamless integration with database systems and the interpreted host language.

By the end of Year 1 of the project, we have investigated the following specific research problems.

A) RIOT-DB. We believe that many tried-and-true ideas from database research, such as external-memory algorithms, pipelined execution, and cost-based optimization, may be applied in the RIOT setting. To test these ideas, and to demonstrate the feasibility of applying them in a transparent manner, we built RIOT-DB (CIDR 2009), which automatically maps data and computation in R to an underlying database system. R's extensibility features allow us to define alternative data types for vectors, matrices, and arrays, which are actually stored and managed in a database. We also provide alternative implementation for generic operators such as multiplication, aggregation, etc. When applied to our custom-defined data types, these operators are translated into database view definitions capturing the computation (without actually executing them). This trick allows RIOT-DB to build up, one operator at a time, bigger view definitions for complex R expressions. When the result is finally needed (e.g., for output), the SQL query defining the view is optimized and executed by the database system, which automatically provides pipelined execution, I/O-efficient algorithms, and high-level, inter-operation optimization. RIOT-DB shows it is indeed possible to achieve I/O-efficiency transparently, not only because R is a high-level language, but also because we can extend it in a clean way to enable inter-operation optimization. On the other hand, RIOT-DB also reveals gross inadequacies of generic database systems in handling array data and statistical computation, such as storage and execution overhead, and lack of advanced algorithmic and optimization support for common numerical operations. Despite these inadequacies, RIOT-DB's database-style execution and optimization allow it to significantly outperform R over large datasets, demonstrating the promise of our approach. Having served its purpose, RIOT-DB is no longer under development; the code is made available (as is) to the public on the project Web site.

B) RIOT. Based on the lessons learned from RIOT-DB, we are building the next generation of RIOT, which features a lightweight data management layer that is highly specialized for statistical computing and embedded in an extension to R. We seek to apply the successful ideas from database research in this new context, without the overhead and inadequacies of existing database systems. For data that already resides in a database system, we also seek to optimize processing across the RIOT/database boundary to enable automatic push down of computation closer to data for efficiency. More specifically: 1) RIOT includes a custom storage manager for arrays, providing advanced layout options and optimization for sparsity that may vary across space and time (more on this in (C) below). 2) Relational query operators turn out to be an inappropriate abstraction for many

statistical and numerical operations. Not only is it awkward to express such operations, but it is also difficult to optimize the resulting relational queries. We develop a new expression algebra for RIOT that models both relational and linear algebra operations, allowing capture of statistical computation in a high-level and intuitive fashion amenable to further optimization. 3) RIOT allows users to analyze data in databases without 'glue' code. Users focus on what data to pull out of the database instead of how to do it; RIOT automatically determines what queries should be pushed down to improve efficiency. 4) For expression and database push-down optimization, we develop a combination of heuristics and cost-based techniques. Heuristic, rule-based transformations are simple to extend and quick to apply. More sophisticated optimization incurs more overhead, but the potential saving justifies its use for expensive expressions. To this end, we are developing a comprehensive cost estimation framework. As an integral part of optimization, we need to make decisions on data layout, intermediate result materialization, timing of evaluation, and place of evaluation (database vs. RIOT). We also need to be aware of numerical stability issues, and provide sufficient control for programmers to intervene when they arise. 5) The implementation of pipelined execution in database systems carries significant CPU overhead, which we cannot ignore in statistical computing workloads. The new RIOT execution engine should apply compiler techniques such as loop fusion, array contraction, and just-in-time compilation to eliminate this overhead. A prototype of RIOT was demonstrated in ICDE 2010, which included (1) and partial realization of (2)-(4). At this point, a limited number of physical operators have been implemented; operators that support sparse array options are still lacking. Using a simple cost model, the prototype optimizer was able to make decisions on operator ordering, database push-down, and data layout. Numerical stability issues have not been considered. The execution was still element-at-time. Hence, the RIOT system is still under active development.

C) RIOT native store. Previous studies have found databases inadequate in storing arrays, especially dense ones. The main reason is that the relational model does not exploit the fact that arrays are ordered collections of data; storing array indices for a dense array wastes storage and only slows down access. As for a sparse array, it is unnecessary to store the zero-valued elements that occupy most of the array; storing only the nonzero elements together with their array indices is more efficient. An index such as a B-tree can be used for fast access to elements. Even though database systems are better at handling sparse arrays, it is still difficult to support various array layouts, i.e., linearization of elements of multidimensional arrays on disk with proper clustering to facilitate efficient access. Since dense and sparse arrays require dramatically different storage strategies, we need a storage manager flexible enough to handle matrices of arbitrary sparsity. Another complexity is that array sparsity may vary over time and over different regions of the array. For example, an initially sparse matrix may be gradually populated with new data until a subregion becomes dense. These problems motivates the design of a novel array storage manager for RIOT, with the following goals: 1) It should control data layout so that it is efficient for different access patterns; it should also provide the option of optimizing for a pre-defined access pattern. 2) It should achieve high space utilization for both dense and sparse

arrays, and those whose sparsity varies. This is desirable because densely packed arrays take less blocks of I/O to read. An interesting observation made in the development of the RIOT storage manager is that we can exploit the finite, discrete index domain of array data to develop better index update strategies. This work is currently in progress.

D) SSD-based storage. SSDs, based on flash hardware, have shown promise as an alternative to magnetic disks because their good random read performance and power efficiency. However, in-place updates are expensive as they require erasing and rewriting large blocks of data. Several efficient indexing schemes have been proposed for SSDs, most of which rely on batch reorganization to avoid random in-place updates. So far, none of these schemes consider concurrency control, which limit their practicality because batch reorganization can severely impact the performance of concurrent data accesses. We are developing an index structure for SSDs that support efficient concurrent accesses. This work, currently in progress, will build the foundation for incorporating SSDs for efficient processing in RIOT.

The progress we have made thus far, as summarized above, are in line with the project plan outlined in our original proposal. In Year 2, we plan to wrap up work on (C) and (D), and continue to refine and implement the next generation of RIOT, focusing on developing a more efficient execution framework, better optimization, and support for sparse matrix operations. We have also established collaboration with HP Labs on the RIOT project. HP Labs will be providing additional funding to RIOT in Year 2 through their Innovation Research Program. The PI will be visiting HP Labs in Beijing in July 2010 to help jump-start the collaboration. The collaboration will focus on the issues of database extensibility and automatic data parallelization. For database extensibility, we will identify target applications, design, implement, and evaluate a framework for extending databases for large-scale statistical analysis, and demonstrate its seamless integration with RIOT. For automatic data parallelization, we will identify target applications and explore parallelization opportunities at both GPU and cloud level via pilot studies and prototyping activities.

In terms of educational activities, in Spring 2010, the PI taught a graduate-level course titled 'Database and Programming Languages: Crossing the Chasm,' which covered research results and themes related to RIOT. During Year 1, the PI has also supervised undergraduate researcher Weiping Zhang, who contributed to the development of RIOT's native storage manager. Since May 2010, the PI has been working with undergraduate researcher Jiaqi Yan, who is studying several information retrieval applications (PLSA and manifold ranking) with scalability demands.

Findings:

In Year 1, we have invested most of our effort in the prototyping and development activities. Experience gained from these activities has provided us with valuable insights and ideas for planning out the reminder of the project. Published results so far include: 1) the RIOT 'vision' paper in CIDR 2009, which also covers RIOT-DB; 2) a demonstration of a prototype for the next generation of RIOT in ICDE 2010. We have released the code for RIOT-DB on the project website.

Training and Development:
Ph.D. students: Yi Zhang, Risi Thonangi, Herodotos Herodotou.

Undergraduate students: Weiping Zhang, Jiaqi Yan

These students have gained research experience in algorithms, compilers, databases, high-performance computing, and programming languages.

Outreach Activities:

Journal Publications

Books or Other One-time Publications

Yi Zhang, Herodotos Herodotou, and Jun Yang, "RIOT: I/O-Efficient Numerical Computing without SQL", (2009). Conference Paper, Published
Collection: Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR '09)
Bibliography: Asilomar, California, USA, January 2009

Yi Zhang, Weiping Zhang, and Jun Yang, "I/O-Efficient Statistical Computing with RIOT", (2010). Conference Demonstration Description, Published
Collection: Proceedings of the 26th International Conference on Data Engineering (ICDE '10)
Bibliography: Los Angeles, California, USA, March 2010

Web/Internet Site

URL(s):
http://www.cs.duke.edu/dbgroup/Main/RIOT
Description:

Other Specific Products

Contributions

Contributions within Discipline:
At the end of Year 1 of the project, we have made a series of solid contributions toward enabling efficient statistical analysis over massive datasets. We have built two functional prototypes, RIOT-DB and RIOT, and published in CIDR 2009 and ICDE 2010. For detailed descriptions of these contributions, please refer to the section of this report on research and education activities. More importantly, experience from our prototyping and development activities has provided values insights and ideas.

Contributions to Other Disciplines:
The PI is part of two other interdisciplinary projects---one funded by NSF, which studies how to collect and analyze ecological data from a sensor network, and another funded by NIH, which develops analytical and modeling tools for immunology. Some of the work in this project is

motivated by the problems faced in the other projects, which will in turn benefit from our research results. As RIOT matures, it will be made available to the general statistical computing community for broader impact.

Contributions to Human Resource Development:

Ph.D. students: Yi Zhang, Risi Thonangi.

Undergraduate students: Weiping Zhang, Jiaqi Yan.

Contributions to Resources for Research and Education:

The PI has integrated the research being carried out under this project into the classes he is teaching (see the section of this report on research and education activities for details). The RIOT tools are also publicly available in open source to other researcher and educators.

Contributions Beyond Science and Engineering:

Conference Proceedings

Special Requirements

Special reporting requirements: None

Change in Objectives or Scope: None

Animal, Human Subjects, Biohazards: None

Categories for which nothing is reported:

Activities and Findings: Any Outreach Activities

Any Journal

Any Product

Contributions: To Any Beyond Science and Engineering

Any Conference