CSR:Small:Exploring Timing-Energy Tradeoffs on Heterogeneous Computing Platforms

Project Summary

Summary.

Intellectual merit.

Broader impacts.

Keywords:

1 Introduction

In recent years, the microprocessor industry has turned to heterogeneous multi-core processor designs for the next generation of embedded systems. By integrating relatively slower, lower-power processor cores with relatively more powerful and power-hungry ones, it is possible to dramatically improve energy efficiency while achieving high performance. Such heterogeneous computing platforms are seeing widespread adoption in many cyber-physical systems (CPS), which are more sophisticated and intelligent computing systems that closely interact with the physical world, including advanced automotive systems, medical CPS, and smart robotics. For example, the ARM big.LITTLE, which is a heterogeneous computing architecture integrating relatively slow and fast cores, has been widely adopted in the automotive systems [**?**]. This heterogeneous computing technology has been used to serve multiple automotive needs such as infotainment as well as advanced driver assistance systems, and is shown to be able to "deliver exceptional power and performance that aligns to the vision of scalable solutions for automotive customers [**?**].

Two Fundamentally Conflicting Goals of Heterogeneous Computing in CPS: Timing Correctness and Energy Guarantees. A major challenge of reliably adopting heterogeneous multicore processors in CPS such as automotive systems is the need to ensure predictable real-time correctness (i.e., enabling timing constraints to be analytically validated at design time). The functional correctness of a CPS hinges crucially upon the temporal correctness, as the control operations depend on the processing of certain environmental sensing and computation tasks. Establishing real-time correctness requires real-time resource allocation methods, whose focus is on judiciously allocating resources to various tasks so that all the required timing constraints can be satisfied. Although traditional real-time resource allocation methods can ensure timing correctness, they often fail to make or mostly ignore energy guarantees, which is another critical goal of equal importance for many CPS due to the stringent size, weight, and power (SWaP) constraints imposed by such systems (even with sub-components powered by batteries). Energy guarantees would be critical for safety reasons of many systems and in general beneficial for system designers who have a fixed energy budget and want to support the maximum amount of workloads that require real-time correctness within that budget.

It is quite challenging to simultaneously consider the two goals of achieving real-time correctness and energy guarantees, because they are fundamentally in conflict. On one hand, real-time resource allocation methods often need to maximize the resource utilization and make greedy scheduling choices to guarantee timing correctness for the worst case. On the other hand, however, methods that yield energy guarantees typically require scheduling decisions to be energy-efficient for the current case, implying that in some cases it is wise to scale down the voltage and frequency of certain cores or even force them to idle.(?Cong: Best to incorporate Hank's data support and citations herein.?) A good real-time resource allocation algorithm may yield excessive amount of energy or thermal hotspots on the many-core chip; while energy-aware methods often fail to make real-time guarantees. Besides this challenge, the heterogeneity among cores can greatly complicate resource allocation because "choices" must be made when selecting the core upon which a task will execute. When jointly considering timing correctness and energy efficiency in the presence of such heterogeneity and dynamic computing needs, resource allocation becomes even more interesting but complicated due to the huge design space of timing and energy tradeoffs. Resolving these issues will be the focus of this proposal.

Proposed approach. Most existing works on real-time resource allocation in heterogeneous computing systems focus on guaranteeing timing correctness (e.g., see [**?**, **?**, **?**] for an overview). A few recent works [**?**] focusing on exploring the timing/energy tradeoff space in a heterogeneous computing system made the following critical assumption: all processor cores can operate at max speed all the time while sustaining safely. This assumption unfortunately invalid due to the critical "dark silicon" problem, i.e., essentially all processors are over-provisioned and have much larger max compute capacity than they can safely sustain (this has never been true for embedded systems before but it is now [**?**]). The thrust of this research is to

simultaneously achieve the goals of timing correctness and energy guarantees on heterogeneous computing platforms containing processor cores with varying speeds by answering the following research questions: **(*i*) how to guarantee timing while running processor cores as slow as possible to respect given energy budgets? (*ii*) how to quickly detect and avoid timing errors by utilizing the over-provisioned processing capacity in short bursts?** Our proposed research is fundamentally motivated by the observations that (*i*) it is significantly more energy efficient when slowing down processor cores (compared to the race-to-idle strategy), and (*ii*) processor cores cannot operate at max speed all the time and are (sometimes significantly) over-provisioned for what they can sustain safely. (Sec. 2 will provide detailed data support for these observations.)

?Cong: the above paragraph needs more work, as I feel I have not concisely and clearly shown the unique angle that was considered in existing works. That invalid assumption made in existing works needs to be re-worked.?

We intend to show that the fundamental timing/energy tradeoffs can be explored by leveraging recent research by the PI's group that has led to a new set of *optimal* resource allocation algorithms for uniform heterogeneous multicore-based systems containing processor cores with varying speeds [**?**, **?**, **?**, **?**, **?**]. These algorithms can analytically guarantee fast and analytically bounded response times for complex workloads implemented using rather general graph-based formalism and executed in a heterogeneous computing system. We will consider such algorithms as the basis for determining the best configurations of resource allocation strategies and processor cores' dynamic voltage and frequency scaling (DVFS) settings, which are most energy efficient for a given set of workloads. Significant further development is needed that considers the dynamic and heterogeneous performance and energy characteristics exhibited on the processor cores. Since there are numerous choices that can be made regarding resource allocation algorithm, the task-to-core mapping strategy, and DVFS settings, the potential solution space for this problem is vast. Tasks can be allocated for CPU resources globally (i.e., a task can be mapped onto any resource) or via partitioning (a task can only be mapped onto a designated resource). Also, task priorities may be either static or dynamic. The efficiency of DVFS-incurred energy saving can also be different for cores with different speeds. In this project, we propose to carefully evaluate the various alternatives in the space of potential solutions on top of real hardware and determine which configurations are preferable are given workloads. An real implementation-based empirical evaluation is thus a focus of this project.

Research objectives. We will pursue the following four-step plan to accomplish our research goal.

- **Step1: Identify the most energy-efficient configurations of resource allocation and DVFS for a given workload that guarantee timing:** We will first design workload mapping and resource scheduling algorithms for processing workloads on a heterogeneous multicore processor. The goal is to guarantee timing while minimizing energy consumption by running processor cores as slow as possible. For each devised algorithm, formal timing validation tests will be developed.
- **Step 2: Detect and avoid timing errors.** Although slowing down cores may be most energy-efficient, it may cause timing errors more easily (e.g., due to sudden workload bursts). Thus, we will further develop robust mechanisms for quickly detecting and avoiding timing errors. Our main idea is to achieve this goal through exploiting the over-provisioned processing capacity in short bursts.
- **Step 3: Carry out overhead-aware schedulability studies.** The research in the first two steps will provide solutions for simultaneously achieving timing correctness and energy guarantees on heterogeneous computing platforms. To evaluate their effectiveness in practice, we will incorporate them in real hardware (using the ARM big.LITTLE architecture) and conduct large-scale overhead-aware schedulability experiments with both synthetic workloads with widely varied parameters and benchmarks. We plan to apply an extensive methodology that is designed for comparing real-time resource allocation strategies in an overhead-cognizant way [**?**, **?**, **?**], which is proven to be effective for many real-time application systems.

- **Step 4: Conduct case-studies.** To determine if our proposed methods can be applied in practice, we intend to conduct case-study evaluations using several specific real-time workloads seen in automotive systems. For example, one such application we will consider is real-time object recognition, which is heavily performed in automotive systems for implementing tools such as collision avoidance and traffic sign recognition. We will evaluate the performance in terms of several specific metrics for using one or more preferable configurations identified in Step 3 to support such workloads.

Qualification. The PIs are well-qualified to conduct this research. The proposing team has worked on various aspects of real-time systems, heterogeneous multicore computing, and energy optimization in the past years, ranging from theoretical real-time analysis [**?**], real-time operating systems [**?**], heterogeneous multicore architecture [**?**], and energy efficient computing under various computer architectures [**?**]. These efforts have resulted in a number of publications accepted by several systems- and architecture-related premium conference and journal venues such as SOSP, RTSS, ISCA, ASPLOS, IPDPS, Micro, PACT, and IEEE Transactions on Parallel and Distributed Systems. PI Liu has been working on developing device driver and OS support for heterogeneous real-time systems accelerated by co-processors such as GPUs [**?**, **?**,**?**,**?**,**?**,**?**,**?**,**?**], resource allocation on heterogeneous platforms [**?**], and real-time schedulability analysis [**?**, **?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**,**?**]. PI Hoffman at the University of Chicago has rich experience in ?INPUT FROM HANK?

2 Background and Related Work

In this section, we first describe the heterogeneous computing architecture and the associated energy and performance characteristics of such platforms. ARM's big.LITTLE will be used as an example heterogeneous computing platform we choose to focus in this document. To ensure timing correctness, real-time resource allocation techniques must be used. We will thus discuss traditional real-time scheduling-related terminology and workload models. Finally, we describe the experimental platform to be used in our research. Due to space constraints, our overview of prior work focuses on research of direct relevance to our agenda.

2.1 Hardware Platform

2.2 Real-Time Scheduling Algorithms and Schedulability Tests

The goal of real-time scheduling is to guarantee timing predictability; in other words, every task is schedulable if it meets the predefined timing constraints at design time. Being "schedulable" depends on whether task deadlines are hard or soft. For a hard real-time (HRT) task, its deadline must be met; while for a soft real-time (SRT) task, missing some deadlines can be tolerated. For example, in an automotive system, autonomous control and emergent collision avoidance are HRT tasks; while automatic lane following and traffic sign recognition can be viewed as SRT tasks. SRT constraints can be defined in several ways. We assume a well-studied SRT notion [**?**, **?**, **?**, **?**, **?**, **?**] that a SRT task is schedulable if its response time can be provably bounded. (Such bounds would be expected to be reasonably small.) In this research, we consider both HRT and SRT possibilities.

To ensure timing predictability, we must develop two kinds of algorithms: *real-time scheduling algorithms* and *schedulability tests*. A scheduling algorithm (i.e., scheduler) determines when to execute which task and on which resource. Scheduling algorithms can usually be divided into two major categories: (*i*) partitioning, and (*ii*) global scheduling. Under partitioning, tasks are statically partitioned among processors [**?**,**?**,**?**,**?**,**?**,**?**]. Different processors can apply different scheduling algorithms. A partitioning algorithm example is partitioned earliest-deadline-first (EDF), which uses the EDF algorithm as the per-processor scheduler. Under

EDF, jobs with earlier deadlines have higher priority. In contrast to partitioning, under global scheduling, a single global ready queue is used for storing ready jobs [**?**, **?**, **?**, **?**]. Jobs are allowed to migrate from one processor to another at any time. A global scheduling algorithm example is global EDF (GEDF), under which jobs are EDF-scheduled using a single ready queue. Clustered scheduling is hybrid of partitioned and global scheduling in which tasks are statically assigned to a cluster of processors, among which the task can freely migrate.

Depending on the scheduling algorithm being used, at design time a corresponding schedulability test verifies whether a task system will be schedulable, provided that the system behaves according to its parameterized specification. Finding an optimal scheduling algorithm that experiences no capacity loss (i.e., 100% resource utilization) is not always possible. If a non-optimal scheduling algorithm is used, then a schedulable task system may be deemed unschedulable by the corresponding schedulability test due to algorithmic capacity loss. Our goal of designing energy-efficient scheduling algorithms and schedulability tests is to minimize the overall capacity loss.

Workload Models. Real-time embedded systems commonly perform recurring operations. For example, the object recognition application commonly involved in implementing many autonomous-driving assisted functionality is naturally recurrent. Specifically, the camera sensor captures an image at a certain frequency; each image is compressed and then analyzed using certain object recognition algorithms, and is expected to be completed by the invocation of the next frame; e.g., each invoked job ideally should complete the corresponding workloads within 33*ms* for a 30 frame-per-second camera.

Generally speaking, a specified real-time system consists of a collection of recurring real-time processes called "tasks". A widely studied general model of recurrent workloads is the *sporadic task model* [**?**]. In this model, the specified system is composed of a collection of recurring[1] sequential tasks, $\tau = \{\tau_1, ..., \tau_n\}$. Each such task τ_i releases a succession of *jobs* $J_{i,1}, J_{i,2}, ...$ and is defined by specifying a *period* T_i, a *relative deadline* D_i, and a *worst-case execution time (WCET)* C_i. Successive jobs of τ_i are released at least T_i time units apart, and one that is released at time t has a deadline at time $t + D_i$. Also, each such job executes for at most C_i time units. Each job released by any task has a *response time* that defines the duration from its release to its finish time. A task τ_i's response time equals the maximum job response time among all of its released jobs.

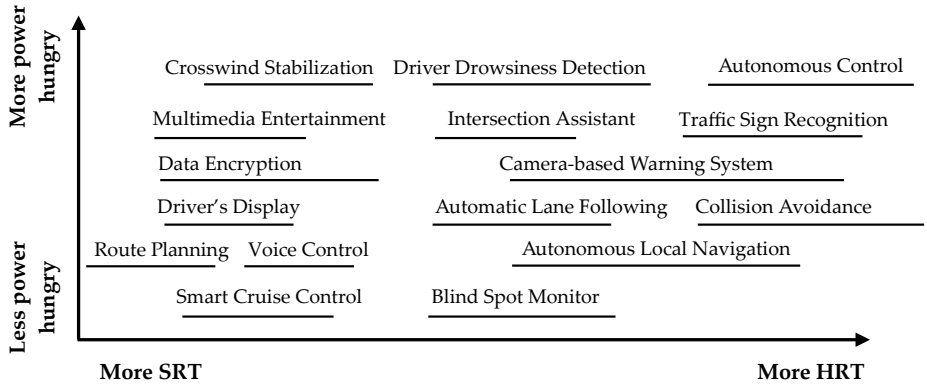

Figure 1: Spectrum of possible timing and energy requirements for a number of automotive workloads.

Automotive workloads. Heterogeneous computing platforms may be applied in a number of application domains where a wide range of timing and energy constraints are required. A particularly focused application domain we will investigate in this research is advanced driver-assisted automotive systems, where workloads with various timing and energy requirements are seen [**?**], as illustrated in Fig. 1. The

[1] Although our focus in this document is on the notion of recurrence found in the sporadic model, we intend to consider other such notions in our work.

automotive system represents an application domain where using heterogeneous computing platforms to achieve real-time performance and energy efficiency is particularly compelling. In such systems, cores with different performance and energy characteristics can be allocated to workloads with different timing requirements. For example, more powerful cores can be used to process HRT workloads such as traffic sign recognition and collision avoidance; while less powerful but more energy efficient cores can be used to process SRT workloads such as voice control and route planning. Dosing so may yield significantly less energy consumption compared to homogeneous platforms because less power-hungry workloads do not need to be processed by powerful cores, which results in necessarily high performance at the cost of energy loss. Due to these reasons, heterogeneous computing platforms such as our targeted ARM's big.LITTLE have been seeing widespread adoption in automotive systems [**?**].

Real-time OS. The OS platform in this project will be an open-source real-time Linux extension called LITMUS^{RT} [**?**]. The PI was involved in developing this testbed [**?**, **?**]. The main purpose of LITMUS^{RT} is to provide a useful experimental platform for applied real-time systems research. Currently LITMUS^{RT} extends Linux (version 3.10.5) by implementing a set of common scheduling algorithms as plug-in components. Numerous publications [**?**, **?**, **?**, **?**, **?**, **?**, **?**] have shown LITMUS^{RT} to be a very valuable testbed for experimentally evaluating real-time scheduling algorithms by considering real system overheads.

2.3 Related Work

Real-time scheduling in heterogeneous systems.

Exploring timing-energy tradeoffs in heterogeneous systems.

3 Detailed Research Plan

3.1 Step 1: Identifying Energy-Efficient Configurations that Guarantee Timing

3.2 Step 2: ...

3.3 Step 3: Implementation and Overhead-Conscious Schedulability Studies

3.4 Step 4: Conduct Case Studies

3.5 Timeline

4 Broader Impact of the Proposed Work

Cong Liu

Department of Computer Science, University of Texas at Dallas, Richardson, TX 75081
http://www.utdallas.edu/~cong/

Professional Preparation

Wuhan Univ. of Technology	Computer Science	B.S. with honor, 2005
Auburn Univ.	Computer Science	M.S., 2008
Univ. of North Carolina at Chapel Hill	Computer Science	Ph.D., 2013

Appointments

Assistant Professor, Department of Computer Science, University of Texas at Dallas, 9/13-present.

Products

(i) Most Relevant Publications

- Husheng Zhou and Cong Liu, "*Task Mapping in Heterogeneous Embedded Systems for Fast Completion Time*", Proceedings of the 14th ACM International Conference on Embedded Software (EMSOFT), 2014.
- Guangmo Tong and Cong Liu, "*Supporting Soft Real-Time Sporadic Task Systems on Heterogeneous Multiprocessors with No Uilitzation Loss*", IEEE Transactions on Parallel and Distributed Systems (TPDS), 2015.
- Husheng Zhou, Guangmo Tong, and Cong Liu, "*GPES: A Preemptive Execution System for GPGPU Computing*", Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015.
- Cong Liu, Jian Li, Wei Huang, Juan Rubio, and Evan Speight, "*Power-Efficient Time-Sensitive Mapping in CPU/GPU Heterogeneous Systems*," Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), pp. 23-32, 2012.
- Guangmo Tong and Cong Liu, "*Supporting Read/Write Applications in Embedded Real-time Systems via Suspension-aware Analysis*", Proceedings of the 14th ACM International Conference on Embedded Software (EMSOFT), 2014.

(ii) Additional Products

- Jianjia Chen, Wenhung Huang, and Cong Liu, "*K2U: A General Framework from k-Point Effective Schedulabiliy Analysis to Utilization-based Tests*", Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS), 2015
- Jianjia Chen and Cong Liu, "*Fixed-Relative-Deadline Scheduling of Hard Real-Time Tasks with Self-Suspensions*", Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS), 2014.
- Cong Liu and Jianjia Chen, "*Bursty-Interference Analysis Techniques for Analyzing Complex Real-Time Task Models*", Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS), 2014.
- Cong Liu and James Anderson, "An O(m) Analysis Technique for Supporting Real-Time Self-Suspending Task Systems," Proceedings of the 33th IEEE Real-Time Systems Symposium (RTSS), pp. 373-382, 2012.
- Cong Liu and James Anderson, "Task Scheduling with Self-Suspensions in Soft Real-Time Multiprocessor Systems," Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS), pp.425-436, 2009, **Best Student Paper Award.**

Synergistic Activities

Service to the Scientific and Engineering Community

- TPC, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015, 2016.
- TPC, IEEE Real-Time Systems Symposium (RTSS), 2014, 2015.
- TPC, Euromicro Conference on Real-Time Systems (ECRTS), 2016.

Collaborators and Other Affiliations

(i) Collaborators

Dr. Evan Speight, IBM Research Austin Lab;
Dr. Jian Li IBM Research Austin Lab;
Dr. Tian He, University of Minnesota, Twin City;
Dr. Jianjia Chen, TU Dortmund (German);
Dr. Yu Gu, Singapore University of Technology and Design (Singapore);
Dr. Wei Gao, University of Tennessee, Knoxville;
Dr. Shinpei Kato, Nagoya University (Japan).

(ii) Graduate and Post-doctoral Advisors

Dr. James Anderson, Professor, Department of Computer Science, University of North Carolina at Chapel Hill.

(iii) Thesis Advisor and Post-graduate Scholar Sponsor

Mr. Guangmo Tong, PhD Student, Computer Science, UTD (since 2014)

Mr. Husheng Zhou, PhD Student, Computer Science, UTD (since 2014)

Ms. Xia Zhang, PhD Student, Computer Science, UTD (since 2014)

Mr. Yuchuan Liu, PhD Student, Computer Science, UTD (since 2015)

Mr. Gbadebo Ayoade, Masters Student, Computer Science, UTD (since 2013)

Facilities, Equipment, and Other Resources

General computing environment. The Department of Computer Science hosts more than 40 research labs with access 24 hours a day, 7 days a week; 2 open labs for graduate students; and a 128-seat open access lab for undergraduate students. These labs consist of over 500 state-of-the-art, high-performance workstations and high-end PCs, all connected via gigabit Ethernet switches with a redundant fiber up-link to provide fast access to the campus network and the Internet. Nine classrooms and one large lecture hall with the latest computer and audio-visual equipment are available. Academic coursework, project, and computing systems are comprised of Linux x86_64, Windows Server, and Solaris 10 executing on a collection of physical servers and virtual server private clouds. The servers are connected via fiber channel and iSCSI to a thin-provisioned 14TB 3PAR mesh-active storage array.

Specific computing facilities. Computer Science Open Access Lab: This lab is accessible to all students currently enrolled in any course in the department of computer science (128 seats). The lab consists of 128 Dell OptiPlex 980 machines with i7-870 Processor (8M Cache, 2.93 GHz), 4GB RAM, 22" (Dell P2210) flat-panel display, Windows 7, 64-bit; Two HP Laserjet 9040 printers are also located in the lab. The lab is open from 8am to 11pm, M-F, and 11pm-7pm Sat/Sun. The Lab is maintained by the CS Tech Staff. A monitor (typically a graduate student hired half-time) is on premises all the time.

Graduate Windows Lab: This 15 seat lab is accessible to all graduate students. Equipment consists of Dell Precision T1650 computers with Intel XEON CPU E3 1225 v2, @3.20GHz, 8GB RAM, 24" flat-panel monitors, Windows 7 Enterprise and an HP Laserjet P4015n printer. The lab is accessible 24/7 with smart card access.

Project Design Lab: This 20 seat lab is accessible to all CS undergraduate/graduate students who take senior design project class. It consists of Dell Optiplex 760 machines with Intel Core2 Duo, E8400@3.00GHz, 2GB RAM, 17"flat-panel monitors, Windows 7 Enterprise. The lab is accessible 24/7 with smart card access.

General Access Systems: The Department also maintains several systems in its server room to which remote access is provided. These include the Network Programming System that are ccessible to all CS students enrolled in network programming and parallel processing courses (45 systems); Students access these systems via SSH; All these 45 systems run Linux CentOS 6.2 x86_64. The Department also maintains two large Linux compute servers (Linux CentOS 6.2 x86_64, Dell PowerEdge R710 w/2 six-core 2.53GHz Intel Xeon, 32GB RAM) and two large Solaris computer servers (Sun Fire V440 w/4x1GHz UltraSPARC-IIIi, 8GB RAM).

Server clouds: The Department also maintains a general purpose server cloud, a VMware vSphere 4 operating on 5 ESX hosts with dual 6-core processor 128GB RAM servers, used for teaching the cloud computing related courses as well as a network security courses server cloud, a VMware vSphere 4 executing on 4 processors of 6-core Opterons, 128GB RAM, and four 4Gb redundant fiber channel SAN connections.

In addition to the above labs and facilities, CS students also have access the UTDesign Studio that is equipped with modern computing facilities and equipment with more than 30,000 square feet of dedicated space, where students and corporate partners can create, innovate, design, build and learn. Further information can be found at http://www.utdallas.edu/utdesign/corporate/studio/. The UTDesign studio is primarily used by undergraduate seniors taking the capstone project course.

A 30′ x 20′ lab is available for Dr. Liu's students. The lab can comfortably accommodate 8 students and the computing equipment. Eight workstations equipped with state-of-the-art NVIDIA GPUs currently in the PI's lab will be available to the project.

Data Management Plan

This data management plan covers all of the data products to be produced in this project. It includes all source code that implements the ecosystem of GPU resource management, any source code added to LITMUS^{RT}, the suite of case-study programs, and scripts used to test and evaluate the implementation of the proposed components, along with the data collected during evaluation efforts.

Source code. All source code will be freely available for direct download from public web servers maintained by the UTD Computer Science Department under an open source GPL license. Additional case-study test programs and scripts will be freely available under the open source BSD license and directly downloadable from public web servers maintained by the UTD Computer Science Department. Our intend in making these data products freely available is that other researchers can verify evaluation results, and relevant research conducted by other interested groups can be facilitated.

Evaluation data. Evaluation data (including raw data and resulting graphs) can be several terabytes in size. It is thus currently not feasible to make this data available to the public through direct download. We will store and back up such data on secured servers maintained by the UT-Dallas Computer Science Department. Interested parties may request copies of this data through making arrangement for private direct download or using the postal system.

Privacy. No personal data will be acquired in this project. There are no privacy concerns for the storage of data.

Long-term storage. All data products produced in this project will also be archived in the UTD computer science computing repository for permanent storage.