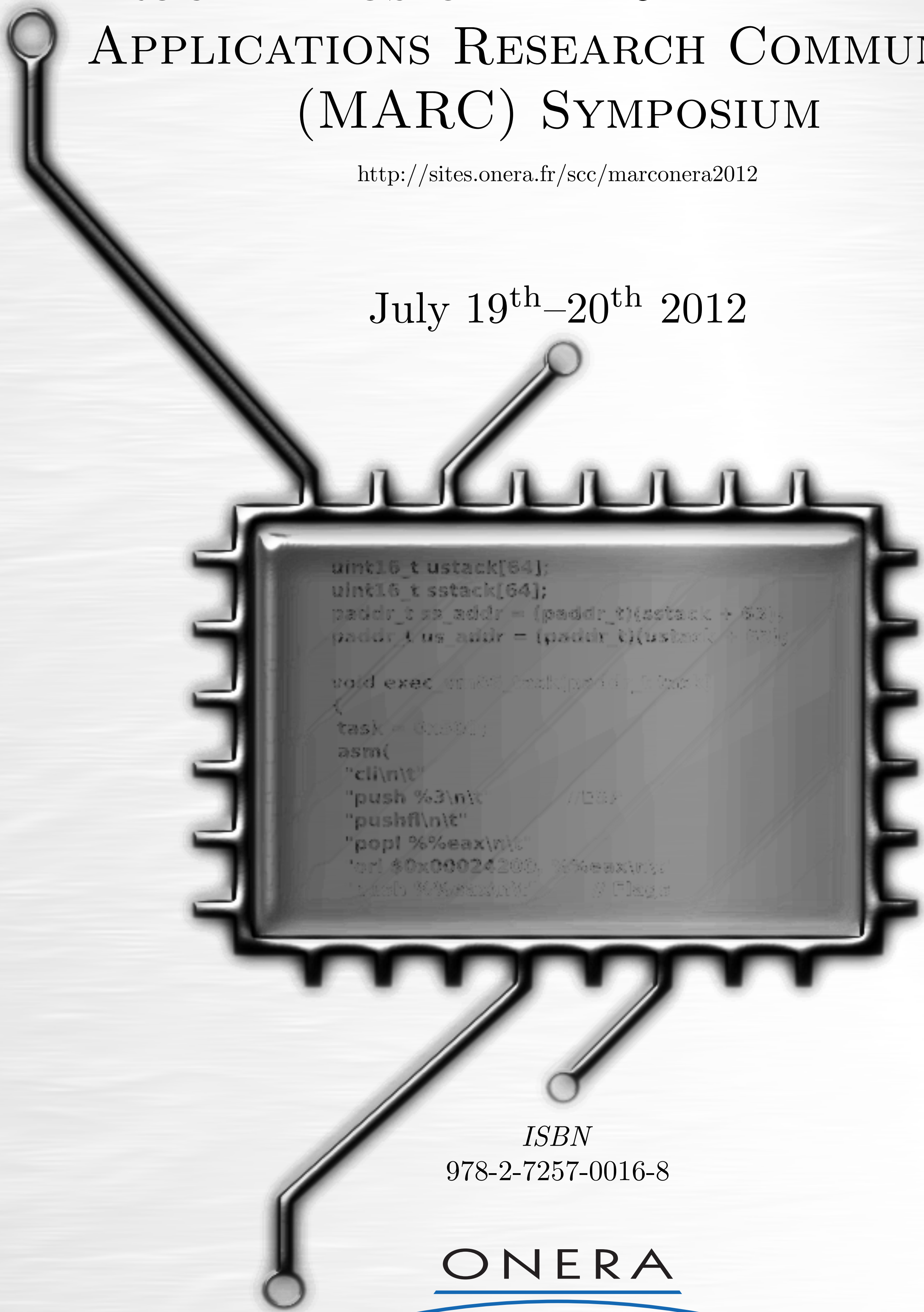
Proceedings of the 6th Many-core Applications Research Community (MARC) Symposium

http://sites.onera.fr/scc/marconera2012

July 19th–20th 2012



```
uint16_t ustack[64];
uint16_t sstack[64];
paddr_t ss_addr = (paddr_t)(sstack + 63);
paddr_t us_addr = (paddr_t)(ustack + [illegible]

void exec_[illegible]
{
 task = [illegible]
 asm(
  "cli\n\t"
  "push %3\n\t            //[illegible]
  "pushfl\n\t"
  "popl %%eax\n\t"
  "orl $0x00024200, %%eax\n\t"
  [illegible]
```

ISBN
978-2-7257-0016-8

ONERA

THE FRENCH AEROSPACE LAB

Enabling Computation Intensive Applications in Battery-Operated Cyber-Physical Systems

H.C. Woithe, W. Brozas, C.Wills, B. Pichai, U. Kremer
Dept. of Computer Science
Rutgers University
Piscataway, NJ 08854
{hcwoithe, wbrozas, cwills, bsp57, uli}@cs.rutgers.edu

M. Eichhorn
Institute for Automation and System Engineering
Ilmenau University of Technology
98684 Ilmenau, Germany
mike.eichhorn@tu-ilmenau.de

M. Riepen
Intel Labs Braunschweig
Braunschweig, Germany
michael.riepen@intel.com

***Abstract*—Autonomous underwater vehicles (AUVs) have become indispensable tools for marine scientists to study the world's oceans. Real time examination of mission data can substantially enhance the overall effectiveness of AUVs in oceanography. However, current AUV technology only allows a detailed analysis of data after completion of a mission. The ability to perform on-board analysis of real time data is computationally intensive, requiring an energy efficient programming infrastructure that can be adapted to battery operated, energy constrained vehicles.**

Intel's 48-core SCC system exposes a collection of performance and energy/power knobs that can be refined for dynamically changing computation vs. energy tradeoffs. In this paper, we illustrate the potential benefits of these knobs for environment modeling and path planning. These applications are important for any autonomous cyber-physical system. Our experimental case study targets AUVs, particularly the Slocum glider. The results show that selecting different core, network, and memory controller speeds have a significant impact on the overall performance and energy requirements of our applications. Furthermore, the best selection is non-trivial and will depend on the available energy and computational needs of other mission critical tasks executing concurrently with modeling/path planning applications.

I. INTRODUCTION

Cyber-physical systems (CPSs) monitor events and conditions in the physical world, process information acquired through different sensors and input devices, and then determine a set of possible actions in response to the observed events and conditions. These systems often rely on battery power for long periods of time, so energy-aware data acquisition, data processing, and actuation is an important issue for increasing the lifetime and/or effectiveness of the overall system. There have been two major developments in recent years that have influenced the design and use of such autonomous systems; (1) the arrival of multi-core and soon many-core systems in the context of battery operated devices, and (2) the development and deployment of a variety of sensors and input/output (I/O) devices for such systems. Both trends are related since additional sensors or I/O require more computational power, and more computational power enables the use of additional, or more sophisticated sensors or I/O.

Multi-core systems have emerged that are designed to work in battery-operated devices. Such multi-core platforms provides substantial computational capabilities at low energy costs, making the execution of applications possible in autonomous environments that people did not believe possible only a few years ago. Recently, ARM announced the big.LITTLE system to improve energy efficiency of high-performance mobile platforms [1] while Intel has produced an experimental 48 core system called the SCC [2]. This system is not commercially available and has been designed to enable scalability research, particularly in the context of energy-aware computing. Allowing its cores, on-chip communication network and memory controllers to be dynamically configured with respect to supply voltage and/or frequency gives the SCC a wide range of performance and energy tradeoffs to support energy-efficient executions of mission critical applications.

In this work, we use ocean modeling (ROMS) and path planning applications for a buoyancy-driven autonomous underwater vehicle (AUV), the Slocum glider, to illustrate how the energy-aware features of the SCC could be used to react to changing energy vs. performance tradeoff requirements. Changes to these requirements can be triggered by the computational needs of other applications which are considered more mission critical resulting from the observation of an internal or external event. Clearly, avoiding obstacles has high priority when navigating through a busy shipping lane. Encountering a physical phenomenon like an algal bloom could also trigger the use of additional sensors and data processing applications. In addition, later phases of a long duration mission may have to deal with reduced battery power and energy budgets, putting more severe constraints on the applications that can be effectively executed. Our case study shows that

1) there are different performance/tradeoff points,
2) finding the best performance/tradeoff point during a mission is non-trivial, and also may change for different parts of a mission and their power caps, and
3) deploying a system such as the SCC in a battery-operated environment like an AUV can provide crucial computational capabilities.

Systems like the SCC could also be deployed in propeller-driven or hybrid AUVs in addition to buoyancy-driven gliders.

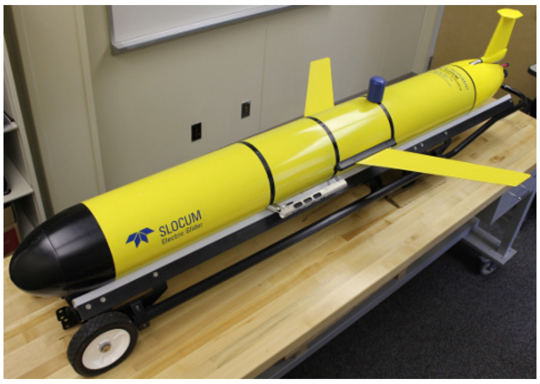


Fig. 1. One of our Slocum glider autonomous underwater vehicles equipped with a double payload bay and an acoustic modem. The shown configuration is 180 cm long and weighs about 90 kg.

Typically, propeller-driven systems have a significantly larger energy budget and a maximal mission duration of days to weeks, rather than weeks to months as the Slocum glider.

II. Slocum Glider

The Slocum glider belongs to a class of autonomous underwater vehicles that make use of a buoyancy engine, instead of a propeller, to traverse the ocean. The vehicle is a commercial product, manufactured by Teledyne Webb Research [3]. Buoyancy-driven flight, for the Slocum, is accomplished through the movement of a piston at the front of the vehicle. Retracting the piston causes the vehicle's displacement of water to decrease, thus allowing the glider to dive. Conversely, extending the piston increases displacement and enables the AUV to climb. Flight pitch can be fine-tuned through the movement of an internal battery pack. Along with wings and a controllable fin, the glider is able to navigate through the ocean at approximately 0.35 m/s [4]. A Slocum glider AUV on a benchtop is shown in Fig. 1.

The success of the glider as a research platform will depend on how well it can satisfy the increasing demands of ocean scientists for more and increasingly complex sets of sensors. Being capable of dynamically reacting to phenomena *in situ* is also becoming more common place. These requisitions necessitate increased computational capabilities. Current stock Slocum gliders are only equipped with two 16 MHz Persistor computers [5], one designated for the flight of the vehicle, while the other collects scientific data from sensors. In previous work, [6] several Linux single board computers (SBCs) were integrated and deployed within the AUV to track and dynamically adjust the vehicle's flight profile to fly within a thermocline off the coast of New Jersey. A system such as the SCC could be used in place of one of these SBCs to provide a flexible architecture that can effectively balance computation and energy requirements.

III. Applications

Additional computational capabilities can save energy by more effectively managing the use of energy expensive sensors. The SCC is particularly well suited for this task because multiple energy saving algorithm/programs can run simultaneously on the chip, each with their own power and energy characteristics and tradeoffs. This section will provide an overview of such applications.

Dead Reckoning - Localization is a critical challenge for underwater operations. Typically, collected sensor data is tagged with spatial and temporal coordinates. AUVs can use GPS localization while at the surface, and dead reckoning (DR) while diving. Unfortunately, DR can result in significant localization errors in the presence of underwater currents. A Doppler Velocity Log (DVL) can be used to remedy this situation by performing bottom tracking, which allows the vehicle to measure its relative speed, thereby improving DR. However, operating the DVL sensor itself, and processing the acquired data can be energy and computation intensive. Without reliable localization many scientific missions are not feasible, including under-ice deployments where acquiring a GPS position at the surface is not possible.

Sensor Triggering - The Slocum glider does not currently support fine-grained or cross-sensor adaptive sampling. Sensors are typically turned on all the time, or active only on dives or climbs. The effectiveness of some sensors can be improved by making them part of a trigger chain, where low cost sensors activate more costly, but more precise sensors. Adaptive sampling may require significant physical modeling efforts and data processing capabilities.

ROMS - The Regional Ocean Modeling System (ROMS [7]) comprises a traditional ocean forecast model complemented by advanced variational data tools that allow the assimilation of 4-dimensional data, and more importantly, the sensitivity of the forecast to the present and future ocean state and the observational sampling pattern. For example, ROMS can be used to help optimize the path a glider takes between waypoints, or to indicate the regions where new observations would lead to the greatest improvement in forecast precision.

Charting the 3-dimensional and time varying pattern of these anomalies in ocean temperature and salinity represents an attractive test-bed for integrating ocean observation and simulation through adaptive sampling and smart control on a single platform. Optimizing the integrated system will necessitate trading off the sampling frequency, the sensors that are active, the distance traversed by the AUV, the ocean model computational effort, and communication, all of which make demands on the available battery power and energy.

Path Planning - The task of the path planning algorithm presented in this paper is to find a time-optimal path from a defined start position to a goal position while evading all static as well as dynamic obstacles in the area of operation, with consideration of the dynamic vehicle behavior and the time-varying ocean currents. The path planning algorithm, named the Time Variant Environment (TVE) algorithm [8], is based on a modified Dijkstra Algorithm [9]. A time-variant cost function is calculated during the search to determine the travel times (cost values) for the examined edges of the graph. This modification allows the determination of a time-optimal path in a time-varying environment [8]. Proper path planning can be crucial if an AUV must arrive at a target location to observe a short lived phenomenon. It can also save time and energy since it allows the vehicle to navigate through strong ocean current fields.

IV. EVALUATION

As previously discussed, the SCC is particularly well suited for parallel applications. For this reason, the ROMS and path planning programs are targeted in our investigation to marry a CPS with a parallel capable infrastructure, like the SCC, that can provide the necessary knobs to tradeoff power and energy restrictions with application runtime deadlines.

For our evaluations, we generate custom settings for the SCC. These settings initialize the SCC with different core, network and memory configurations. Because we envision multiple programs communicating at the same time on the SCC, we also studied non-standard mesh network speeds in the hope that it could provide us with additional insights on the tradeoff space of the SCC.

Both ROMS and the path planning application make use of RCKMPI [10] for message passing which should provide comparable performance to RCCE [10], [11]. The MPD process manager, recommended for use with the SCC, is used throughout the experiments. All cores boot a Linux 3.1.4 kernel image. Finally, power measurements were gathered from the SCC infrastructure.

A. ROMS benchmark

We evaluated the feasibility of running ROMS on the SCC using a sample benchmark provided with ROMS. The benchmark consists of 512x64x30 grid points and 200 time step iterations. The main computation is a two-dimensional stencil with nearest-neighbor communication. The grid is divided into tiles, where the total number of tiles must match the number of cores that are part of the computation. The grid's tile dimensions were chosen to maximize the size of grid points calculated per core and to reduce the size of the halo/ghost regions. Larger halo regions require more communication and computation. We empirically validated that the tile dimensions used are optimal for the grid size and the number of cores.

Our evaluation of the ROMS benchmark program is shown in Fig. 2. A diverse set of configurations of CPU, mesh, and memory were tested with both 24 and 48 cores. In most cases, the runtime for 48 cores, Fig. 2(b), is lower than the 24 cores (Fig. 2(a)) with the same setting. The fastest configuration with 24 nodes performed nearly identically to the second slowest configuration of 48 nodes, and outperformed the slowest. In scenarios where soft runtime deadlines are acceptable, numerous options and tradeoff points are available for ROMS. A global application scheduler can consider these alternatives during the arbitration of the next SCC setting.

The average power consumption during the execution of ROMS is shown in Fig. 2(c) and Fig. 2(d) for 24 and 48 nodes, respectively. Throughout the experiments, lower mesh speeds reduced power by several watts. The most pronounced effect on power were high tile frequencies. Battery operated CPSs, like the Slocum glider, may need to observe power caps during operation, since actuators and other systems can increase the power load on the device. Therefore, it may not always be an option to run the fastest configuration with the highest node count.

Similar to the runtime, it is generally more energy efficient to use 48 cores instead of 24 cores to run the benchmark. The highest setting for the 24 nodes in Fig. 2(e) is, however, similar to the lowest configuration of the 48 cores seen in Fig. 2(f). When comparing their respective runtimes, the 48 node setting does outperform the 24. Across the figures, the crossover points are very similar and are prospective tradeoffs opportunities. Because of the dynamic nature of AUVs, mission priorities can change often, emphasizing the importance of a suitable arrangement for runtime, power and energy.

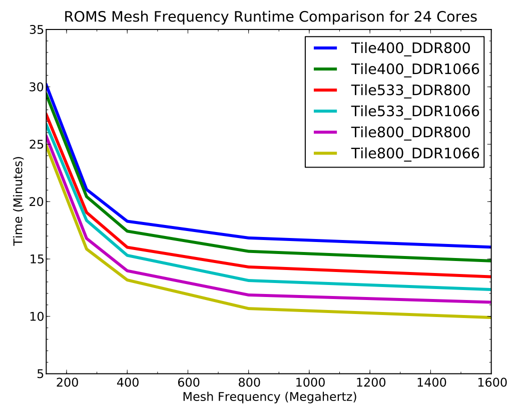
B. Path Planning

We have ported both the serial (S-TVE) and parallel (P-TVE) versions of the TVE path planning algorithm to the SCC. The input parameters to both programs were identical throughout the benchmark tests. Since parameter choice can have an impact on the amount of parallelism the program is capable of during execution, we have chosen a set of parameters consistent with our previous work [12].

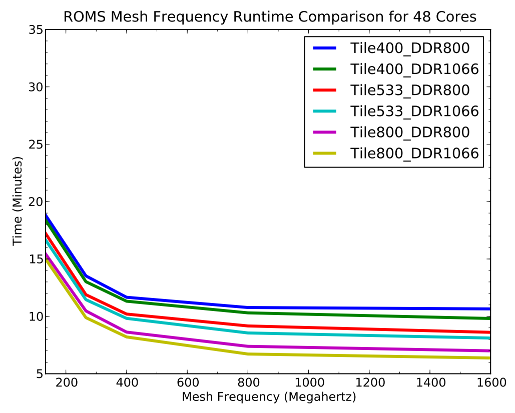
The opportunity for parallelism that was exploited and implemented in P-TVE was to find the optimal dive profile depths for the vehicle. Because the AUV can experience different currents at various depths, it may be advantageous for the vehicle to glide within a certain depth range for portions of the flight. For each edge in the graph, this dive profile calculation is evaluated for 20 distinct depths ranges.

Results of the path planning programs for the SCC configurations are show in Fig. 3. S-TVE results are only available for one core since there is no parallelism involved. P-TVE has a master/slave architecture where the master delegates work to slaves that perform the dive profile task, so at least two cores are required. The MPI-NOOP results measure the overhead of the MPI infrastructure. It is a modified version of P-TVE which initializes MPI and immediately exits.

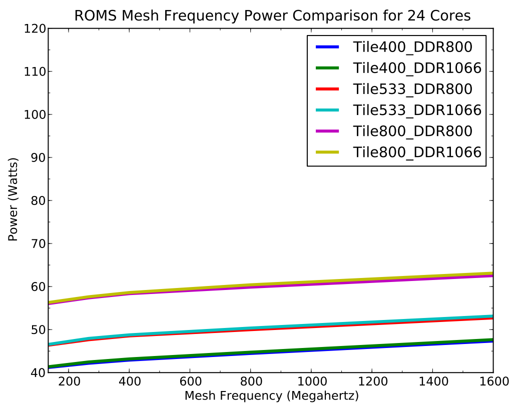
The program runtime, Fig. 3(a), and dive profile search time, Fig. 3(b), decreased as the number of cores increased for P-TVE. There is an initial communication overhead for two cores, when compared to S-TVE, as the master must delegate work to the slave. The step-wise behavior is explained by the number of iterations of work delegation that is performed by the master. For example, with 11 cores, 10 slaves perform work for two work iterations. In the case of 12 cores (11 slaves), the second work delegation will leave one slave idle. Because of the input parameter of 20 distinct depth range calculations, the optimal number of nodes should be 21. This accounts for one master with 20 slaves doing one iteration of work. Additional nodes only provide overhead in P-TVE as indicated by the



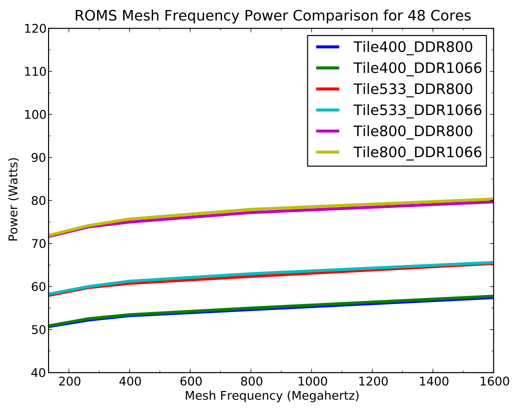
(a)



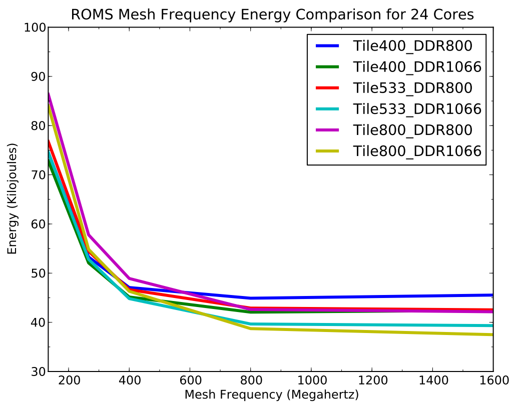
(b)



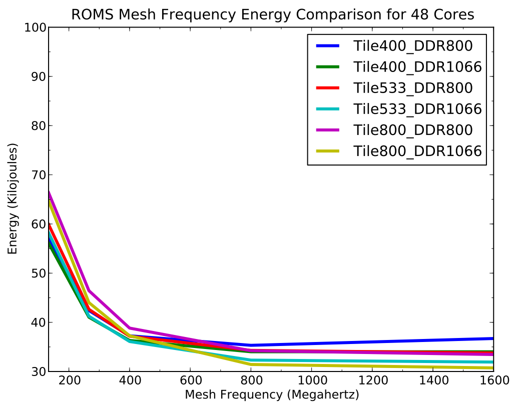
(c)



(d)



(e)

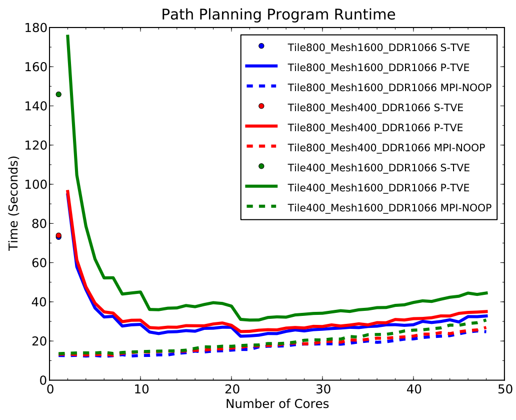


(f)

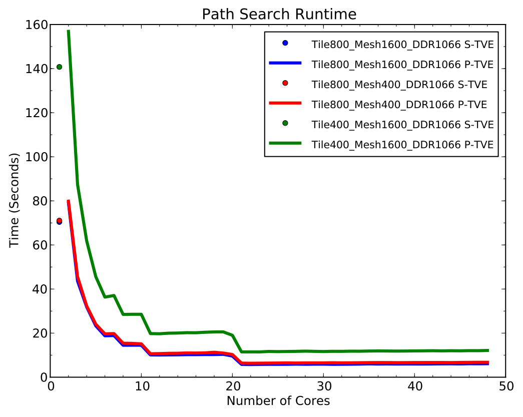
Fig. 2. ROMS evaluation results for various SCC settings. The execution times for ROMS using 24 (a) cores and 48 (b) cores. The average power , (c) and (d), of the SCC during the execution of program. The energy required to run ROMS for 24 (e) and 48 (f) cores.

speedup of the dive profile search in Fig. 3(c). The speedup for each setting is normalized to the S-TVE search time of the same setting. If the number of profile searches is increased, additional cores could be used with a concomitant increase in benefit. Additional details are available in [12].

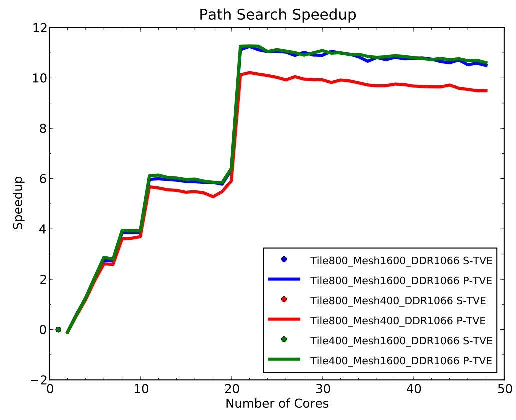
To reduce the power and energy of the program, idle slaves are instructed by the master to enter into sleep mode. In sleep mode, a slave performs an asynchronous receive call instead of a blocking receive call. This allows the slave to sleep in



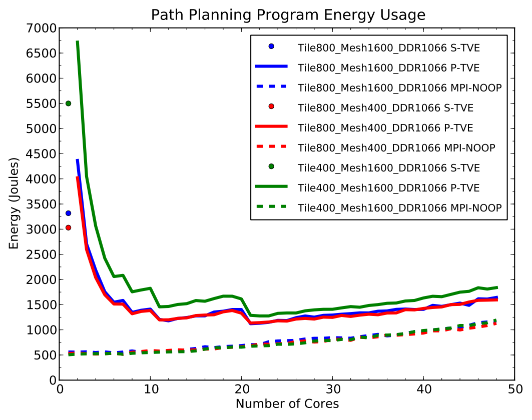
(a)



(b)



(c)



(d)

Fig. 3. Evaluation results for the path planning program for various SCC settings. The runtime of (a) is the time required for the entire program to execute. The search time, (b) is the time required to perform the search for the optimal dive profile. Speedups for each of P-TVE is relative to the S-TVE with the same SCC setting. The energy required for the entire program execution (d) is based on (a).

between update checks of the asynchronous call. Although this introduces latency for the first receive, it greatly reduces the overall energy used by the slave. This latency is evident in Fig. 3(c), especially when there are a high number of idle slaves. For example, after 21 nodes, even the idle slaves that will never perform any work experience the latency because they wait for the termination message to be sent by the master.

The evaluation indicates that the path planning program is more reliant on computation than communication as the slowest core speed setting has the longest program and profile search runtimes. Lowering the mesh speed does decrease the speedup of the parallelization because it delays communication between the master and its slaves. However, the effect it has on runtime is not as significant as observed when changing the CPU frequency.

The energy required for the planning programs are depicted in Fig. 3(d); it shows opportunities for tradeoffs that could be used when choosing an SCC configuration that will run several programs simultaneously on the chip. Although the runtime of the P-TVE is generally longer for the low mesh speed configuration, the power saved by reducing the mesh frequency translates to a comparable energy profile of the highest speed SCC setting. After 21 cores, even the slowest tile setting could be considered, as the energy difference is not substantial. Similar to the runtime results, energy is wasted on idle slave cores. We hope to address this issue in the future.

C. *Discussion*

The applications described, along with others, could be required to run simultaneously on the SCC. Depending on the current needs of the system the priority of tasks may change periodically, or change based on observations of phenomena in the environment.

Power caps can also restrict the selection of high power SCC settings. A Slocum glider typically uses alkaline battery packs, so the supply voltage drops as energy is consumed. A glider's fresh alkaline battery pack is rated as 1800 watt hours, while the SCC's power demands can range from 40 W to 80 W for our applications. As a comparison, the buoyancy engine of the glider operates at 60 W or more during inflections at 200 m depths. The vehicle must maintain a minimum voltage level at all times to operate safely. The use of actuators, like the buoyancy engine, and sensors, such as a DVL, will increase the power needed by the AUV. It may not be possible to run the SCC concurrently with some sensors, while other sensors can be active at the same time as the SCC provided that the chip does not exceed its allotted power.

Having knowledge of the tradeoff points for an application is critical when choosing a configuration setting. For example, at some point in a deployment, a vehicle's path may need to be resolved rather quickly. Ideally, the highest tile, mesh and memory speed (Tile800_Mesh1600_DDR1066) should be chosen and P-TVE is run on 21 nodes. However, there could be a loose deadline to perform modeling and thus ROMS must also be considered. If the highest setting exceeds the allotted power, a small sacrifice could be made by lowering the mesh frequency. The impact on the runtime and energy of path planning is minimal. While the impact is greater for ROMS, it may still fall within the soft deadline restrictions.

The ROMS tradeoff scenario described in Section IV-A could also be made in the case of a power cap. If there is no need for path planning, and the requirements are such that ROMS should have nearly the same runtime and energy profiles as the best setting for 24 nodes, then the program could be run on all 48 cores at half the tile frequency. This allows the program to not only be more runtime and energy efficient but also greatly reduces the required power. The lowering of the frequency, in this case, is what may be needed to bring the power profile below the cap.

Although we have focused on power cap scenarios, other tradeoff points which concentrate on energy and runtime can be made. This is especially true if more applications, like sensor triggering, are involved in the deliberation. Other cyber-physical systems will have their own hardware and software restrictions and priorities. The SCC can provide CPSs a tradeoff space in which it can make decisions that involve runtime, power, and energy.

V. Conclusion And Future Work

In this paper, we have performed a set of benchmarks for applications on the SCC that could be used in a battery-operated Slocum glider or a battery-operated propeller-driven AUV. The results of our evaluation indicated that the applications expose many knobs for different SCC tile, mesh and memory frequency settings. These knobs can be used to tradeoff program runtime, power, and energy use depending the needs of the AUV and the overall mission. A deployment of our SCC system within one of our Slocum gliders is not possible due to the SCC's particular form factor and system configuration. Our case study shows that future many-core architectures similar to the SCC can play a significant role in making AUVs more effective, autonomous research platforms.

As part of future work, we would like to port our applications to invasive MPI (iMPI) [13]. In particular, based on our evaluation of the path planning application, the overhead of launching MPI processes can be significant using MPD. Similar overheads were observed in [13] and were reduced using iMPI's process manager. We hope that this MPI alternative will help to reduce the energy requirements of applications running on the SCC.

We would also like to extend our evaluation with additional SCC settings. The programs in our evaluation were run in isolation. Performing a similar analysis for multiple applications running concurrently on the SCC could be of interest. Having several programs interacting with the chip may spur interesting effects on the runtime and energy profile of the applications.

Acknowledgments

This research has been partially funded by NSF awards CSR-CSI #0720836 and MRI #0821607, the Murray Visiting Professorship at Rutgers University, and the Internal Excellence Promotion for Research at Ilmenau University of Technology. We are particularly grateful to Intel and the SCC development team for their help and advice throughout this project.

References

[1] A. Peter Greenhalgh, "Big.little processing with ARM cortex -a15 and cortex-a7 www.arm.com/files/downloads/big.little_final.pdf," 2011.

[2] M. Gries, U. Hoffmann, M. Konow, and M. Riepen, "SCC: A flexible architecture for many-core platform research," *Computing in Science Engineering*, vol. 13, no. 6, pp. 79 –83, November-December 2011.

[3] Teledyne Webb Research, "Slocum glider," Falmouth, MA, http//www.webbresearch.com/slocum.htm.

[4] J. G. Graver, R. Bachmayer, N. E. Leonard, and D. M. Fratantoni, "Underwater glider model parameter identification," in *Proceedings 13th International Symposium on Unmanned Untethered Submersible Technology*, 2003.

[5] Persistor Instruments Inc., "Cf1 computer system," Marstons Mills, MA, http://www.persistor.com.

[6] H. Woithe and U. Kremer, "A programming architecture for smart autonomous underwater vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009.*, October 2009.

[7] A. Shchepetkin and J. McWilliams, "The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model," in *Ocean Modelling*, vol. 9, 2005, pp. 347–404.

[8] M. Eichhorn, "A new concept for an obstacle avoidance system for the AUV "Slocum glider" operation under ice," in *Oceans '09 IEEE Bremen*, Bremen, Germany, 11-14 Mai 2009 2009.

[9] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, no. 1, pp. 269–271, 1959.

[10] I. A. C. Ureña, M. Riepen, and M. Konow, "Rckmpi - lightweight MPI implementation for Intel's single-chip cloud computer (SCC)," in *Proceedings of the 18th European MPI Users' Group conference on Recent advances in the message passing interface*, ser. EuroMPI'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 208–217.

[11] T. Mattson, R. Van der Wijngaart, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, and S. Dighe, "The 48-core SCC processor: the programmer's view," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, November 2010, pp. 1 –11.

[12] M. Eichhorn, H. C. Woithe, and U. Kremer, "Parallelization of path planning algorithms for AUVs concepts, opportunities, and program-technical implementation," in *Oceans '12 MTS/IEEE Yeosu*, Yeosu, Republic of Korea, May 21-24, 2012.

[13] I. A. C. Ureña, M. Riepen, M. Konow, and M. Gerndt, "Invasive MPI on Intels single-chip cloud computer," in *Architecture of Computing Systems ARCS 2012*, ser. Lecture Notes in Computer Science, A. Herkersdorf, K. Rmer, and U. Brinkschulte, Eds. Springer Berlin / Heidelberg, 2012, vol. 7179, pp. 74–85.