A Manual Approach and Analysis of Voltage and Frequency Scaling using SCC

Kenneth Berry, Felipe Navarro, Dr. Chen Liu
Department of Electrical and Computer Engineering
Florida International University
Miami, United States of America

Abstract—the current trend of Dynamic Voltage and Frequency Scaling (DVFS) techniques involve algorithms that predict when a processor is in a period of accessing off chip memory and dial down its voltage/frequency during this phase in order to reduce energy consumption with minimal, if any, effect on execution time. These algorithms often operate with a parameter that defines the tolerable performance degradation, because the various operating frequencies that a processor can be set to are often limited. This limit makes it practically impossible to dial down a processor's frequency to the exact optimal frequency that will provide maximal energy efficiency but not affect performance. This leads to a need for these algorithms to include the previously stated parameter to identify cases where choices which degrade performance to an unacceptable level and/or without providing a benefit in energy consumption are avoided. However, the overhead costs incurred by the process of voltage and frequency scaling must also be taken into consideration. We propose a study to determine the impact of these overhead costs on the overall benefit of dynamic voltage and frequency scaling.

I. INTRODUCTION

Our initial approach to conducting this research was to use a program which exhibited varying degrees of computational intensity and run sections of it with different combinations of voltage and frequency to determine which settings were optimal for which sections. With this information we would then run the program again and switch to each optimal frequency for each section of the program. We could then compare the effects of the overhead involved in switching the voltage and frequency with respect to energy consumption and execution time to determine whether the benefit from scaling justified the overhead costs. What is meant by “computational intensity” is processor utilization. Processor utilization indirectly tells us when a program is in a phase of memory intensive operations. It is during these periods that scaling frequency and voltage to a lower setting can improve energy efficiency with minimal effect on performance. In searching for a program which displayed varying levels of computational intensity to use as a basis for this research, we realized that this would incur considerable effort because it required that we developed an intimate understanding of the source code of each program under consideration in order to identify whether or not it exhibited varying levels of computational intensity. Once that was accomplished the sections at which these levels occurred would also need to be defined. We also realized that programs which exhibited this type of behavior most likely used advanced mathematical concepts, making understanding the source code exceedingly difficult. To get around these issues, we decided to create a synthetic program (Here forth referred to as the benchmark program) made up of smaller, more easily understandable programs (Here forth referred to as sub programs). The benefit of this approach was two-fold; Rather than run the entire benchmark program under various frequency and voltage combinations to identify optimal settings for each section, each sub program could be run individually to determine its optimal setting. The second benefit was that the sections of the synthetic program were clearly defined by the start and finish of each sub program. This research was conducted on the Intel Single-chip Cloud Computer (SCC) platform.

II. RELATED WORK

As mentioned before, for the SCC we propose a method of fine tuning a program to have a DVFS implementation to run for many cores. Previous work for such a setup has been mentioned for cluster computing where DVFS programming models are needed. Cluster computers have the problem of increasing performance at the cost of having an increase in power and energy consumption. As mentioned in previous work, the attempts to improve such a problem have been investigated for cluster
computers and such, these techniques can be useful as we move into the many core era. The SCC can be looked as a cluster on a single ship with each tile representing a single computer and thus such methodologies can be useful for future work. Algorithms such as the Beta-adaptive algorithm [1] are useful methods for adding a power-aware feature in cluster computers. If such algorithms can be applied to such systems, then as we move to the many-core era, can be applied as well. Current investigation of power-aware ideologies are not yet available for many cores as since the SCC is the only working model that is currently available. As methods of fine tuning do not current exist for many-cores, our hope that previous methods available for cluster computing can be implemented in future research.

III. APPROACH AND CONSIDERATIONS

Nearly all studies regarding processor power and energy consumption are done on commercially available systems or systems created from commercially available hardware. Accurately measuring the power dissipation of the processor is a challenge in itself. These systems may have functions that can return the systems instantaneous power usage, or the authors may have chosen to measure the systems power through its power source. Some authors have even tapped into the processors power source directly to eliminate the need to estimate the percentage of the system power that is contributed by the processor. Determining power dissipation for a specific program is another issue, considering the processor is performing other tasks such as background services while the test program is running. The SCC does not have a traditional operating system that performs system tasks while simultaneously running user programs. It is effectively idle when there are no programs being run on it. Because of this, power readings taken from the functions provided by the SCC provide an accurate measurement with respect to the execution of a program. We measure the execution time of each program from within the program using the time functions provided by the SCC. Energy consumption is calculated from the product of the interval of time and measured power within that interval. Each sub program used to construct our benchmark program follows a similar structure and basically performs one overall task that is parallelized and split amongst all the cores. Because of this, it is unnecessary to take multiple power readings. The power reading is taken in the middle of the parallelized section to ensure that all cores are not idle at the moment the power reading is taken. This provides us with an accurate energy consumption calculation by limiting the overhead from calling the time and power functions.

The range of voltages and frequencies that can be set on the SCC is rather limited, specifically in terms of the available voltage levels. Because using the same voltage level for a lower frequency will only increase execution time while using similar power, and thereby increase energy consumption, we only use three voltage/frequency combinations to reflect the 3 currently available voltage levels on the SCC. Each subsequent combination uses a higher voltage and frequency than the last. These combinations are specified in Table 1. We ran each sub program using each combination to determine which one provided the best performance in terms of both execution time and energy consumption, using combination 3 as the basis for determination. For example, if combination 2 decreases energy consumption by 20% but increased execution time by 25% as compared with the execution time and energy consumption of combination 3, then combination 3 is deemed to provide the best performance for that sub program. In the case that combination 2 decreases energy consumption by 20% and increases execution time by 20% as well, the combinations are deemed equivalent and the higher combination is used. Although we used combination 3 for comparison, using any of the combinations as a basis for comparison would have produced the same results in terms of determining the optimal combination.

In our benchmark program, the source code of all sub programs were placed sequentially, simulating a larger program with different sections exhibiting different levels of computational intensity. To establish a baseline for execution time, power, and energy consumption, the synthetic program was run using each of the voltage/frequency combinations without scaling any of the sections. We then modified the source code of the synthetic program to scale the voltage/frequency prior to each section to the previously determined optimum setting.

<table>
<thead>
<tr>
<th>Combination</th>
<th>Voltage</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7V</td>
<td>320MHz</td>
</tr>
<tr>
<td>2</td>
<td>0.8V</td>
<td>533MHz</td>
</tr>
<tr>
<td>3</td>
<td>1.1V</td>
<td>800MHz</td>
</tr>
</tbody>
</table>

Table 1. Voltage/frequency combinations used

In our benchmark program, the source code of all sub programs were placed sequentially, simulating a larger program with different sections exhibiting different levels of computational intensity. To establish a baseline for execution time, power, and energy consumption, the synthetic program was run using each of the voltage/frequency combinations without scaling any of the sections. We then modified the source code of the synthetic program to scale the voltage/frequency prior to each section to the previously determined optimum setting.

IV. EXPERIMENTAL RESULTS

We plot the execution time, power, and energy consumption of the benchmark program using all voltage/frequency combinations. The fourth combination shown is using dynamic voltage/frequency scaling for each section of the benchmark program to its optimal setting. As compared with combination 1, combination 2 showed a significant decrease in execution time as well as some decrease in energy consumption. This tells us that combination 1 is simply an inefficient choice compared to combination 2, as it provides no benefit in
either of the criteria we are using to compare performance. This is reflective of our results in
determining the optimal combination for each sub program, as combination 1 was not used in the dynamic
scaling of any of the sub programs for combination 4. As compared with combination 2, combination 4 decreased
execution time by 9.08% but increased energy consumption by 31.87%. As compared with combination
3, combination 4 reduced energy consumption by 17.54% but increased execution time by 28.92%. Both of these
comparisons represent an unacceptable tradeoff between execution time and energy consumption. We attribute this
to the overhead caused by dynamically scaling the voltage/frequency of each section of the synthetic
program. Although the combinations that were scaled to
were the most optimal when the sub programs were run
individually, the benefit provided by these optimal
configurations were overshadowed by the time it took to
switch to them. Each sub program executes based on a
pre-defined number of iterations, and increasing these
iterations substantially could potentially lessen the gap
between the benefit of using the optimal setting for each
section and the extra time required to switch to them. We
also note that the execution time of the synthetic program
was less than that of the sum of all the sub programs
being run individually. This implies that there is
substantial overhead cost in the initialization of the MPI
protocol, as it is only initialized one time in our
benchmark program versus four times in executing the
four sub programs individually.

Figure 1. Execution time of benchmark program for all
voltage/frequency combinations

Figure 2. Average power of benchmark program for all
voltage/frequency combinations

Figure 3. Energy consumption of benchmark program for all
voltage/frequency combinations

V. FUTURE RESEARCH

We plan to continue exploring the overhead costs of
dynamic and frequency voltage scaling using the SCC
platform. We hope to be able to configure our SCC to allow us
more freedom in selecting different voltage/frequency
combinations as well is determine the different overhead costs
from switching to different voltages/frequencies. We also plan
to implement a method to profile the MPI programs we test in
future work to eliminate the need for a quantitative
determination of optimal configuration as we have done in this
paper.

ACKNOWLEDGMENT

We would like to thank our professor and mentor, Dr.
Chen Liu for allowing us the opportunity to work with
the SCC platform and for guiding us through the process
of conducting research. We would also like to thank Intel
for providing us with the SCC.

REFERENCES

