International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842
Issue 12, Volume 3 (December 2016) www.irjcs.com

Scope and Issues in Green Compiler

Premangshu Chanda
Dept. Of Computer Science
St. Xavier's College, Kolkata

Subrata Modak
Dept. Of Computer Science
St. Xavier's College, Kolkata

Pallab Kanti Mukherjee
Dept. Of Computer Science
St. Xavier's College, Kolkata

Shalabh Agarwal
Dept. Of Computer Science
St. Xavier's College, Kolkata

Asoke Nath
Dept. Of Computer Science
St. Xavier's College, Kolkata

Abstract — with the growing demand of electronic devices, efficient conservation of energy has become the major concern. Investments are on the datacentres that consume huge amount of energy at both hardware and software levels as well as maintaining them. In a computing system, design, development and compilation has significant impact on conservation of energy. While it is difficult to control the impact of conservation of energy at hardware level, Green computation and in turn, Green Compilers are safest bet in controlling conservation of energy at software level. Controlled use of resources is achieved by the optimization of compiler and scheduling approach. The focus of this paper is to identify the tools and methods to achieve efficient use of resources and energy consumption and their utilization at compiler level. Distributed Green Compiler is a hardware independent green compiler which distributes source code over the network. Such compilers use green compiling techniques to conserve energy at software levels at compile time by transforming and reshaping binary codes by the application of green strategies. Resources are used efficiently in a manner that consumption of power is minimized and the emission of carbon dioxide is reduced at significant amount. The proposed compiler conserves energy clock cycles for a good range of nearly 40% by applying several green strategies.

Keywords — Distributed Green Compiler (DGC), Energy Aware Compiler (EAC), DIET scheduler, Energy conservation GNU Compiler Collection (GCC).

I. INTRODUCTION

The emergence of Cloud computing has changed the paradigm of computing approach world-wide very rapidly. However, Clouds consume high amounts of energy to host services and applications on their datacenters. Moreover, Global warming has become a major concern which has also imposed its impacts on the use of technology. Green computing attempts to minimize the toxic impact of technology. This can be primarily done by reducing the fuel consumption in computing.

Energy conservation and yielding of high performance has been conflicting while designing green computing strategies. Reduction of logic voltages leads to conservation of energy but the output frequency is significantly reduced, while slower circuits leads to degradation of performance[1]. Multiple low capacity machines can be beneficial over high capacity machines for solving large computationally intensive problems at times when optimization of performance and cost effectiveness is desired. More energy can be consumed by high capacity machines as they do not utilize their resources fully at all times. One way to reduce energy consumption significantly is to implement distributed green strategies over the network. In April 2007, Gartner predicted that the Information and Communication Technologies (ICT) industry produces about 2% of the total global CO_2 emissions [2]. According to a report published by the European Union [3], a reduction in emission of about 15-30% is necessary before the year 2020 to keep the global temperature from increasing below 2^O C. Therefore, satisfactory solutions are essential to ensure environmental sustainability of this new computing prototype [2].

A reliable way to reduce consumption of energy with resource optimization is energy optimization through green computing. Implementation of smaller silicon process geometries, auto-idle detection circuits, or active well-biasing techniques are other methods to optimize consumption of energy at hardware level [4]. The same can be achieved at software level by the implementation of *Green Compilers.* Green Compilers evaluate active processes for energy requirements by the means of program analysis at compile time and reshaping of codes at transformations.

Green aspects can also be introduced during software development cycle such as in software analysis and design [5]. Figure 1 shows the layers of a green operating system [5]. Energy sensitive programs can be injected in Kernel of operating system to achieve higher layers of conservation of energy.

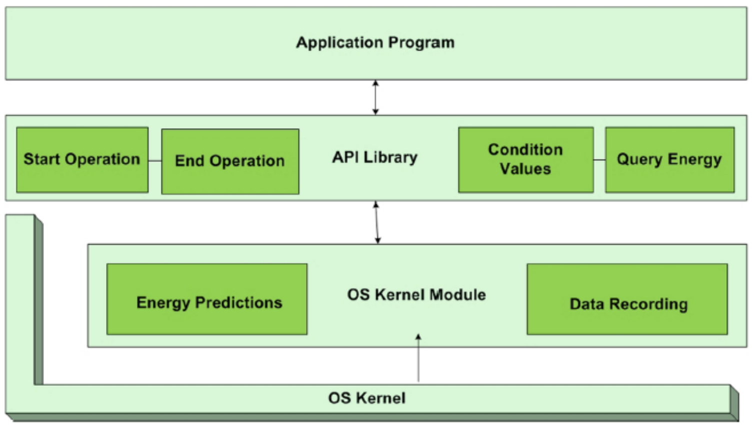


Figure 1 Energy aware computer system architecture.

Compilers can be a good source of software level energy optimization. Hardware independent Green compilers are mostly used in the embedded system. The major focus of this paper is to discuss an energy conservative green compiler to generate an optimized executable of energy conservative and to tradeoff between energy conservation and performance.

DIET clouds has emerged which makes use of DIET architecture [5] providing scheduling at agent level based on user requirements. This section follows a review of related work of green compilers in Section II. Several strategies of energy optimization in green compiler are identified in Section III. Green strategies for software development life cycle are detailed in Section IV. DGC is described in Section V. A brief idea and feature of Green Hill compiler is presented in Section VI. Finally, Section VII concludes the research.

II. RELATED WORKS

Green compilers tend to improve the efficiency of the cloud by affecting clouds as well as datacentres as a whole. Scheduling, Resource management, reducing the amount of active resources executing the workload are focused frequently. Virtualization is achieved in the form of several Virtual Machines which in turn, plays an important role in optimization of efficiency of energy. Though they maximize energy efficiency, but still fail to promise reduction in CO_2 emission. The following are some related works in context to green compiler:

A. Implementing Green strategies to complier

Various energy efficient compilers are active, while some of them are in the process of making. Coffee Compiler for C language is such a compiler that combines software and modified hardware to achieve conservation of energy at compile time [7]. They are primarily built for embedded systems, and they are hardware dependent. For large objects, implementation of such green strategies increases compile time which leads to significant performance degradation.

Another energy aware C compiler is ENCC which was developed at Dortmund University, Germany. The specialty of ENCC lies in the intermediate code which is converted by application of several optimization techniques. ENCC also maintains database about conservation of energy and statistics for each iteration and memory access is preserved.

A distributed open source compiler is MRCC. It uses Map Reduction by preprocessing an input file, it then checks dependency among several source files and determines the safety criterion for compilation, then allocates each on several machines [8]. Though, this is a good advancement to distributed compiling, but the energy conservation criteria is not found to be beneficial and hence, cannot be considered as a green compiler.

Distributed Green Compiler [5] is a hardware independent compiler that implements green strategies on code at software level by reshaping them and through optimization of the source code in energy conservative executable. DGC performs compilation of the programs in several green strategies, some of which are too complex to be handled by energy aware compilers, like the process of recursion elimination. These are essential as if these aspects are skipped, it may lead to decrease in efficiency considerably. DGC aims to target such sensitive areas of the program compilation that are likely to consume more energy and which compilers cannot reshape. DGC facilitates programmers by highlighting such aspects.

III.CARBON AWARE GREEN CLOUD ARCHITECTURE

Carbon Aware Green Cloud Architecture, illustrated in Figure 2 deals with CO_2 emission of Clouds [2]. It consists of three elements: *A third party* which lists all available green Cloud services and their respective energy efficiency, a user or *Green Broker* which accepts Cloud service request and selects a provider, and *a provider* which enables the carbon efficient operation of Clouds. Google has released the energy efficiency of its datacenters [6].

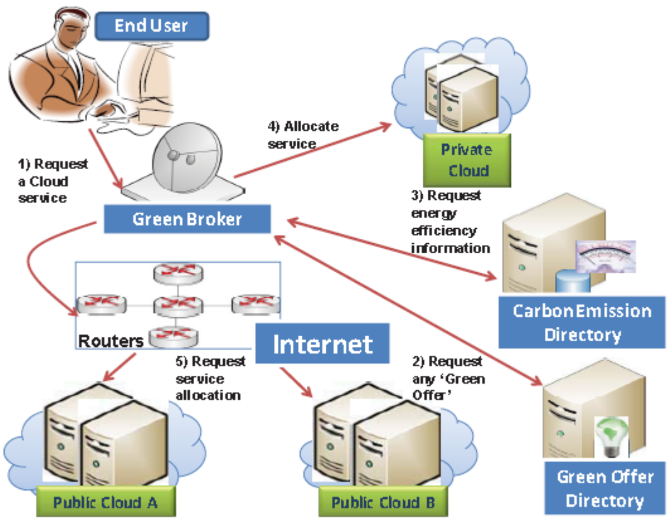

End User
1) Request a Cloud service
4) Allocate service
Private Cloud
Green Broker
3) Request energy efficiency information
Carbon Emission Directory
Routers
Internet
5) Request service allocation
2) Request any 'Green Offer'
Public Cloud A
Public Cloud B
Green Offer Directory

Fig. 2. *Carbon Aware Green Cloud Architecture*

IV.CARBON EFFICIENT GREEN POLICY

Carbon Efficient Green Policy [2] selects a Cloud provider periodically which has minimum carbon footprint. It then initiates Virtual machines to run the jobs. It gains information about the current emission rate of Carbon Dioxide from *Carbon Emission Directory* based on user's request at each scheduling interval. CEGP then sorts the incoming jobs based on Earliest, Deadline First before sorting the carbon footprints. The Carbon footprint of an IaaS Cloud I is calculated. The formal description of the policy [2] is given in Figure 3:

```
while $current\_time < next\_schedule\_time$ do
    RecvCloudPublish(P);
    //P contains information of Cloud datacenters
    RecvJobQoS(Q);
    //Q contains information of Cloud users
Sort jobs in ascending order of deadline;
Sort datacenters in ascending order of $r_i^{CO2} \times \frac{1}{Ieff_i} \times \frac{1}{VMeff_i}$;
foreach $job\ j \in RecvJobQoS$ do
    foreach $datacenter\ i \in RecvCloudPublish$ do
        if $isInitiatedVM(i)$ then
            if $MaxIniVMlimitReached(i)$ then
                Try to schedule the job j on already initiated VMs;
                if $job\ j\ is\ missing\ deadline$ then
                    continue;
                break;
            else
                InitiateVM(i) and schedule job j;
                break;
```

Fig 3: *Carbon Efficient Green Policy*

III. GREEN STRATEGIES FOR COMPILERS

A green compiler analyzes software programs at the time of its execution and reorders the shaping of software source code by applying multiple green aspects during code transformation. Following are some green techniques that can be applied at local, global or inter-procedural level to make program more efficient in terms of energy awareness [5]:

A. USE OF CACHE SKIPPING TO REDUCE POWER CONSUMPTION

A decent approach for green compilation technique is to merely avoid the cache during unnecessary repetition. Loops are the prime targets and of high significance of programming. Though Duplications of loops give high performance, yet they consume more energy.

In the method of cache skipping, the compiler separates the blocks that have very little or no probability of getting executed at next feasible intervals. In ideal situation, this eliminates the need for use of caches [1]. Therefore, power consumption will reduced be significantly.

B. Memory Read / Write vs. Using Register Operands

It has been observed that memory reads and writes costs much more than using register operands. Instructions with register operands have about 300 mA cost per cycle [1]. On the other hand, instructions with memory operands costs 430 mA per cycle for reads and 530 mA per cycle for write. Therefore, it is evident that integrating the use of more register operands significantly reduces running time and eliminates the chances of cache misses.

C. Clustering Instructions

Instruction clustering leads to conservation of energy by running the program as clusters of related or similar signals can be compiled in one run. The research in [9] shows that this method can save energy from 26% to 47%.

D. Instruction reordering and Memory addressing

Energy consumption can be significantly reduced by changing the order in which the program is supposed to be executed. A technique has been proposed using Gray code and Cold scheduling [10] that reduces 20% to 30% instruction switching.

E. Use of Energy Cost database

In the first run of compilation, all possible parse trees are generated and their respective cost is assigned using energy-cost database. In the next run, the tree with the least cost is selected for further compilation.

F. Loop Optimization

There are many loop optimization methods, out of which, *Loop Fission* is the efficient one. In Loop Fission, nested loops are checked across dependency graph in which nodes represent statements and edge corresponds to data dependency. If there is no processing, compiler will generate loop for each statement and run them parallel using interleaved processing.

G. Dynamic Power management

In CMOS, the consumption of power is dynamic, i.e., when the circuit is the state of operation and no power leakage occurs. Dynamic power management system sets the power of its hardware in 'true time' without degrading the performance to decrease probable power waste.

H. Resource hibernation

Hibernation means using low power mode. A compiler reshapes a program behavior using source level transformation in such a way that idleness threshold of a resource can be extended and switched to hibernation mode with less switching.

I. Cloud Aware Task Mapping

Cloud Aware Task Mapping is the use of available services that can be provided by different clouds. Parallel processing is done at host level using cloud services. This technique holds the drawbacks of cost of virtual machine and migration cost of machines, and delay due to network failure.

J. ELIMINATE RECURSION

Since compilers converts recursion into iteration internally, it tends to save some space as compilers execute recursive processes as stack causing degradation of performances and hence, energy consumption.

IV. GREEN STRATEGIES FOR SOFTWARE DEVELOPMENT LIFE CYCLE

At design level, energy can be conserved by making energy efficient structure of software. Following strategies can be used by software developers for making green compilers [5]:

A. Use of Green IDE and Compiler

Several energy aware compilers are available that helps in reducing energy consumption. There are various open source, licensed green compilers such as Green Hill compiler for C and C++, ENCC energy aware compiler for C++.

B. Avoiding Recursion

As recursion uses stack, it takes longer time to execute them which leads to much greater consumption of energy. Therefore, eliminating, or at least avoiding recursion and promoting the use of iteration is a better approach to green compiler [9].

C. USING ENERGY AWARE DATA STRUCTURE

Efficient data structure is more energy conservative. For instance, it has been found that merge sort consumes less energy with arrays than linked list. Using various data structures and using them efficiently leads to efficient green compiler.

D. LESS RUNNING TIME

Any strategies that can reduce the running time of an algorithm will be beneficial. Computing the complexity in Big O notation and reducing it close to linear time complexity is an effective method of compiling in a green compiler.

E. USING GRID AND CLOUD COMPUTING

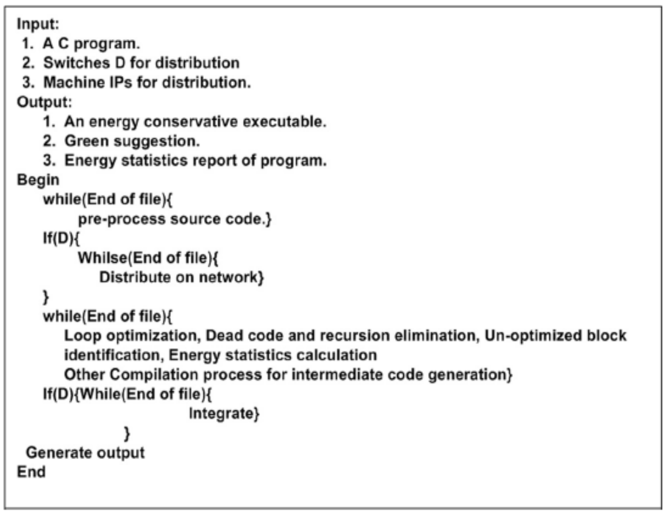
Complex computations that take longer can performed on cloud. For instance, Currency conversion or calculators which are readily available can be made used in programs saving energy, time and hence, cost.

V. DGC: DISTRIBUTED GREEN COMPLIER

DGC [5] is a proposed distributed green compiler. This section provides the basic architecture and high level workflow of DGC.

DGC uses several green strategies during intermediate code conversion to reshape the source code. It applies green techniques in compilation leading to degradation in terms of performance. Although, it is possible to compile program on a single machine, it distributes source codes over a network of physical or virtual machines.

All source codes cannot be reshaped in energy conservative executable like use of register operands, recursion elimination, etc. In such cases, DGC offers green suggestions as it highlights certain areas of the source code, which the compiler fails to reshape for optimization of consumption of energy during intermediate conversion of source code. After successful compilation, DGC provides energy consumption statistics of a program letting the user know how much energy can be conserved in the produced executable.



```
Input:
 1. A C program.
 2. Switches D for distribution
 3. Machine IPs for distribution.
Output:
    1. An energy conservative executable.
    2. Green suggestion.
    3. Energy statistics report of program.
Begin
    while(End of file){
          pre-process source code.}
    If(D){
          Whilse(End of file){
              Distribute on network}
    }
    while(End of file){
        Loop optimization, Dead code and recursion elimination, Un-optimized block
        identification, Energy statistics calculation
        Other Compilation process for intermediate code generation}
    If(D){While(End of file){
                      Integrate}
              }
 Generate output
End
```

Fig 4: Algorithm for DGC compiler [5]

Figure 4 shows the generic algorithm for DGC. The input is a C language program with some optional parameters such as a switch to tell the compiler that program needs to be distributed on network, IP address to specify network machines available for compilation, etc. DGC uses distcc [14], an open source distributed C/C++ compiler using GCC compiler that sends preprocessed source codes over the network and volunteer machines in compilation of the source code.

Figure 5 shows workflow diagram of DGC. The energy cost statistics and green strategies modules require implementation to achieve the functionality of a distributed green compiler. *Green Compilation* and *Output generation* modules of ditcc are modified to perform the mentioned tasks.

DGC gets source files from C source project, which is a collection of source codes and header files. It then pre-processes the files by attaching required header files and libraries. Based on the programmer's preference, the compilers distribute source code over the network or compile them on the same machine. Distribution may be performed by distributing the files on different slaves for compilation. Each request will be sent to first available machine that will process it. Slaves require running cross compiler and a daemon of DGC. In first run of compilation, source code is selected block by block and following *Green strategies* are applied on selected code.

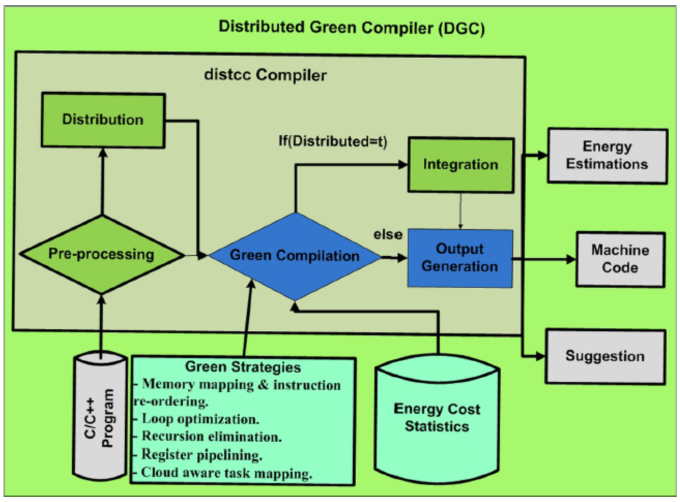


Fig 5: Workflow diagram for DGC

DGC provides own energy conservative data structure for software developers. Following are the features of a DGC compiler:

A. Loop optimization
It is the process to conserve energy during loop execution. Several techniques of loop optimization are available. For example, loop tilling, loop fission, loop fusion and loop unrolling. DGC performs loop unrolling through funroll-all-loops and fvariable-expansion-in-unroller switches of GCC compiler. Funroll-all-loops switches conserves energy by reducing the number of iterations of a loop, while fvariable-expansion-in-unroller copies local variables during loop unrolling for dependency elimination.

B. Dead code elimination
It is the process of removing code resulting in no change in program result. For instance, if a variable is assigned a value and a value is assigned to the variable again. The first assignment is, without a doubt, useless. This saves energy by saving the clock cycles by shrinking the program avoiding implementation of unsuitable operations.

C. Software pipelining
This technique can be used for loop optimization in case of overlapping iterations. GCC compiler supports this feature. In addition to GCC compiler support, DGC uses Modulo Scheduling to perform software pipelining. As these approaches are hardware independent, so is DGC.

D. Elimination of Recursion
Although it is practically impossible to eliminate recursion altogether, DGC converts recursion into iterations as and when possible. If such conversions are not at all possible, DGC suggests green alternatives by highlighting the specific areas of code that falls in this class.

E. Cloud Aware Task Mapping
DGC uses several clusters of virtual machines for distributed compilation process using hardware and software resources to process a compilation problem.

F. Un-optimized code blocks identification
DGC offers green suggestion by highlighting un-optimized blocks of codes of a program. Figure 6 demonstrates that swapping of an array element using memory locations (6(a)) requires more low level instructions than a register swap operations (6(b)) [11].

G. Energy Cost Statistics [5]
DGC maintains energy cost table for each executed instruction in a database. Different parse tree of selected code is generated using Energy Cost Statistics database. Machine code of least energy cost parse tree is generated in the final run. Summarization of reports is done generated at output. Green suggestions aims to highlight areas of program that is not optimized by compiler and can be energy conserved by implementing green strategies discussed in the earlier section. This paper is describing the concept, architecture, and high-level workflow of DGC. The implementation details of DGC are not a part of this research. For the proposed concept, a conservation of 40-60% energy is analyzed if few green aspects are implemented. Suppose a program P uses V_{cc} supply voltage and I average current to achieve the goals within T seconds, then the total energy, E, consumed by a program can be calculated with the equation [12]:

$$E = V_{cc} * I * T.$$

We can write $T = N_{cc} * \tau$, where N_{cc} is the number of clock cycles and τ is the clock period. So, the energy equation can be written as $E = V_{cc} * I * N_{cc} * \tau$. V_{cc} and τ remains same for a specific hardware. Thus they are considered as constants.

The equation becomes $T = N_{cc} * \tau$. Since τ is assumed to be constant, so we need to reduce N_{cc} in order to make a program efficient [5].

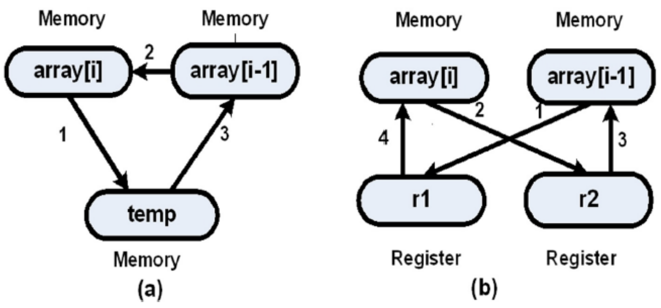


Figure 6: Array element Swap operations

VI. GREEN HILL COMPILER

Green Hill Optimizing Compiler [13] is a green compiler brought to the industry by the Green Hill Company. It is not an open source compiler. The features of Green Hill Compilers are:

A. Fine Tune Output

It uses very careful implementation and cutting-edge technology. Green Hill compiler offers 20% improvement in speed and at least 10% reduction in size compared to other green compilers.

B. Language Support and Tight Integration

Green Hill compilers support various languages such as C, C++, Embedded C++, Ada and FORTRAN, etc. It is built as a part of complete embedded development solution including integrated development environment and real time operating system.

C. Find Bugs Automatically

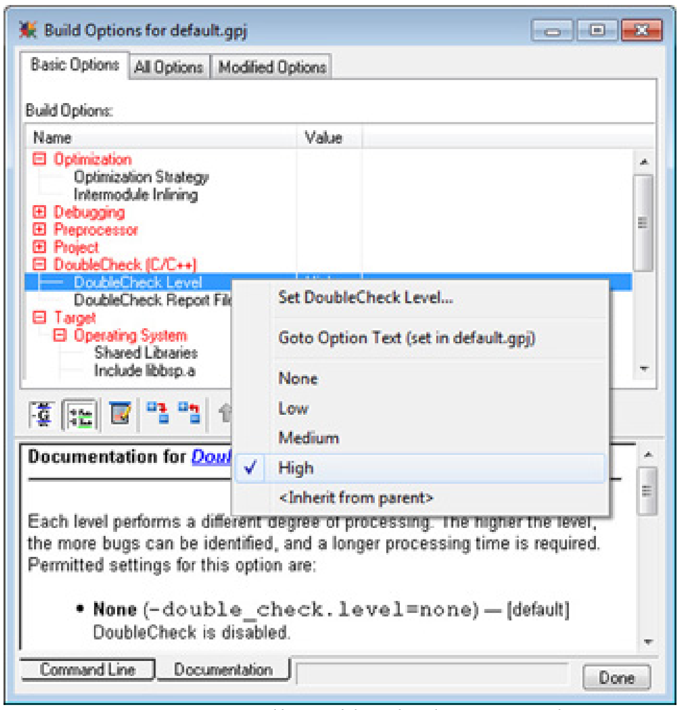
Green Hill Compiler is very efficient in finding bugs by the use of techniques such as Double Check. Moreover, Green Hill compiler's integrated software analyzer also contribute to finding programming errors before running the program saving energy, resources, cost and time.

D. Reduce Processor and Memory Costs

Green Hill compilers reduce memory costs by reducing the size of the executable. Most programs grasp at least 10% improvement relative to GNU compiler.

E. Supports Wide Range of Processors

Green Hill compilers are available for a variety of 32- and 64- bit processors including Power architecture/VLE, ARM, Intel Architecture, ColdFire68K, BlackFin, MIPS, Tricore, etc.



Documentation for Doubl

Each level performs a different degree of processing. The higher the level, the more bugs can be identified, and a longer processing time is required. Permitted settings for this option are:

- **None** (`-double_check.level=none`) — [default]
  DoubleCheck is disabled.

Fig 7: Green Hill Double Check static analyser

VII. CONCLUSION

Green Compiler is a software level technique to conserve energy. A green compiler applies several green strategies to reshape source code during intermediate code conversion generating energy conservative executables. In this paper, several techniques for a good green compiler are highlighted as well as discussing techniques to adopt energy conservative programs. A distributed green compiler is studied that uses some of the identified techniques at compilation levels. Some source codes that cannot be reshaped by the green compiler are highlighted by DGC.

Energy conservation and performance are conflicting goals and compilation time of a program is increased when green strategies are employed. This problem is handled by the DGC by distributing the source code over the network of physical or virtual machines, also providing the option to compile the code on a single node. Performance analysis shows that DGC conserves clock cycles by 30% to 40% when green strategies are employed. The future work of the paper would be to optimize the compiling techniques for better performance analysis and detailed study of performance analysis of DGC with existing compilers, after completion of its prototype.

The provision of resources by cloud has added many advantages in terms of saving cost and effectively use of its resources. One such commercial green compiler called *Green Hill Compiler* is studied and its features are presented in this paper. The said compiler is energy conservative improves speed by 20% and reduces size by 10% by implementing green strategies. The additional feature of several processors, wide language support and separate tool for auto-debugging makes it more efficient and user-friendly. A future study will be to search for prospects and use of green strategies to make the green compilers greener and more efficient so that in addition of efficient utilization of resources, power consumption and Carbon dioxide emission can also be reduced.

VIII. REFERENCES

[1] Bellas N, Hajj IN, Polychronopoulos CD, Stamoulis G(2000) Architectural and Compiler Techniques for Energy Reduction in High Performance Microprocessors. IEEE Trans Large Scale Integration(Vlsi) Syst 8(3):317 326
[2] S . K. Garg, C . H. Yeo, R. Buyya: Green Cloud Framework For Improving Carbon Efficiency of Clouds [Online] Available
[3] Baer, P.: Exploring the 2020 global emissions mitigation gap. http: //www.ippr.org/uploadedFiles/globalclimatenetwork/Exploring_the Mitigation_Gap[1].pdf (Dec 2008).
[4] Bohra, A., Chaudhary, V.: Vmeter: Power modelling for virtualized clouds. In: Proc. of 24th IEEE IPDPS Workshops. Atlanta, USA (2010)
[5] F . Fakhar, B . Javed, R . u. Rasool, O . Malik, K . Zulfiqar: Software level green computing for large scale systems [Online] Available http://www.journalofcloudcomputing.com/content/1/14
[6] Miller, R.: Google: Raise Your Data Center Temperature. http://www.datacenterknowledge.com/archives/2008/10/14/ google-raise-your-data-center-temperature (Oct 2008)
[7] Raghavan P, Lambrechts A, Absar J, Jayapala M, Catthoor F, Verkest D (2008) COFFEE: COmpiler Framework for Energy-Aware Exploration. HiPEAC'08 Proc 3rd Int Conference High Perform Embedded Architectures Compilers 4917:193–208
[8] Cloud Computing., Retrieved at 03. June 2011 [http://fclose.com/b/cloud-computing/article/mrcc-a-distributed-ccompiler-system-on-mapreduce/]
[9] Naik K (2010) A Survey of Software Based Energy Saving Methodologies for Handheld Wireless Communication Devices. Tech. Report No. 2010-13. Dept. of ECE, University of Waterloo
[10] Su C-L, Tsui C-Y, Despain AM (2002) Low Power Architecture Design and Compilation Techniques for High-Performance Processors. Compcon Spring '94, Digest of Papers, pp 489–498
[11] Naik K, Wei DSL (2001) Software Implementation Strategies for Power-Conscious Systems. J Mobile Networks App 6(3)
[12] Hsu C, Feng W (2005) A power-aware run-time system for high-performance computing. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC '05). IEEE Computer Society, Washington, DC, USA, p 1 doi:10.1109/SC.2005.3
[13] Green Hill Compiler features., retrieved on 21, November 2015 from [http://ghs.com/ds/index.php?ds=compilers]
[14] Google Code., Retrieved on 03, December 2015 from [http://distcc.googlecode.com/svn/trunk/doc/web/index.html.]