2011 International Nuclear Atlantic Conference - INAC 2011
Belo Horizonte,MG, Brazil, October 24-28, 2011
Associação Brasileira de Energia Nuclear - ABEN
ISBN: 978-85-99141-04-5

THEWASP Library
Thermodynamic Water & Steam Properties Library in GPU.

Waintraub, M., Lapa, C.M.F., Mol, A.C.A., Heimlich, A.
Waintraub, M., Lapa, C.M.F., Mol, A.C.A., Heimlich, A.
Comissão Nacional de Energia Nuclear, Instituto de Engenharia Nuclear
21941-906, Rio de Janeiro, RJ, Brasil
Waintraub, M. marcel@ien.gov.br **Lapa, C.M.F.** lapa@ien.gov.br
Mol, A.C.A. mol@ien.gov.br **Heimlich, A.** adino@ien.gov.br

ABSTRACT

In this paper we present a new library for thermodynamic evaluation of water properties, **THEWASP**. This library consists of a C++ and CUDA[13] based programs used to accelerate a function evaluation using GPU and GPU clusters. Global optimization problems need thousands of evaluations of the objective functions to find the global optimum implying in several days of expensive processing. This problem motivates us to seek a way to speed up our code, as well as to use MPI on *Beowulf* clusters[5][17], which however increases the cost in terms of electricity, air conditioning and others. The GPU based programming can accelerate the implementation up to 100 times[7] and help increase the number of evaluations in global optimization problems using, for example, the PSO[8] or DE[15] Algorithms. **THEWASP** is based on Water-Steam formulations publish by the *International Association for the properties of water and steam, Lucerne - Switzerland*[2] [16], and provides several temperature and pressure function evaluations, such as specific heat, specific enthalpy, specific entropy and also some inverse maps. In this study we evaluated the gain in speed and performance and compared it a CPU based processing library.

1 Introduction

In many industrial simulators of heat transfer and thermo-hydraulic behavior we need to provide water and steam properties, wich depends on temperature, pressure and volume to correct evaluations of heat transfer or accident simulations. The same occurs in the nuclear field, where we need to provide a fast and reliable method to obtain water properties for application in faster simulations. This paper describes a novel approach to evaluate thermodynamic properties of water and steam publish on Water-Steam formulations by the *International Association for the properties of water and steam, Lucerne - Switzerland*[2] [16]. This new approach is based in Graphics Processor Unit (GPU) and this method employs a two-dimensional six-term Taylor Series expansion [10] around selected grid points on a suitable plane of independent variables. The function evaluations are accelerated by *textures tables* on GPU and employ a massive grid function call to obtain a brute force parallelization. Other mainstream libraries, like ASTEM [11] or NIST-ASME [6] used by RELAP-5 [3], offer a great accuracy but in sequential evaluation and demanding precious time in computation.

1.1 Water Properties as Temperature and Pressure Functions

The water-steam properties defined by IAPWS-IF97 for industrial and general purpose utilization are temperature and pressure dependent and can be classified in five regions as in figure 1. The properties in each region are obtained by derivatives of Gibbs free energy function, defined as

$$G(p,T) = U + pV - TS \tag{1}$$

used in regions 1, 2

where U is the internal energy ($joule$), p is the pressure ($Pascal$), V is the volume (m^3), T is the temperature (K), S is the entropy ($joule/K$) and F is the potential function.

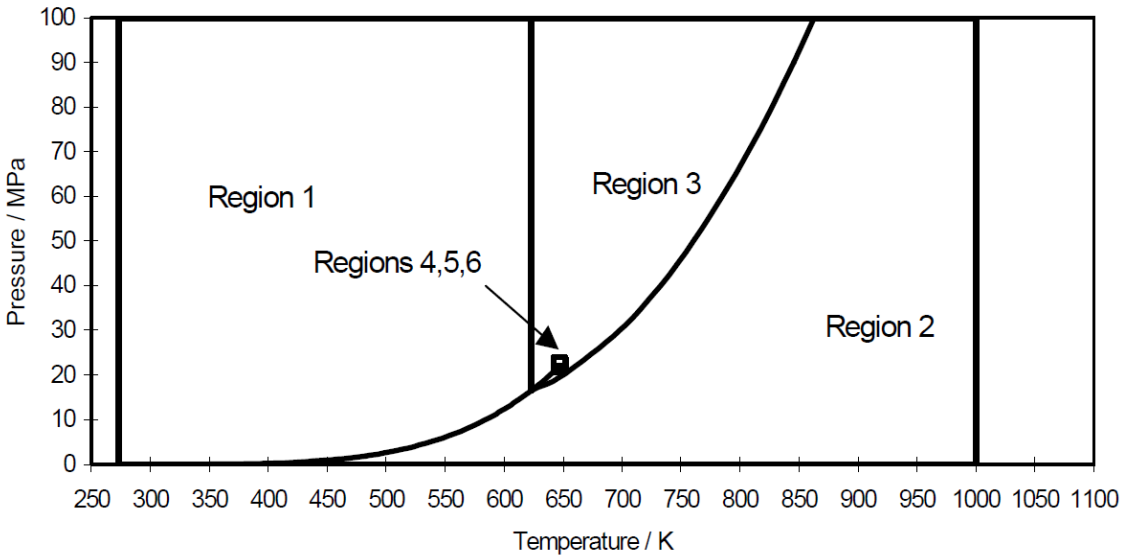

Fig. 1: Regions of IAPWS-IF97

We shortly describe the regions above.

- Region 1: p = 0.0006128 to 100 MPa, T = 275 to 623.15 K, stable liquid (almost equal to Region 1 of IAPWS-IF97)
- Region 2: p = 0.0006128 to 100 MPa, T = 275 to 1000 K, stable vapor (almost equal to Region 2 of IAPWS-IF97)
- Region 4 defines the saturation region: p = 22.064 to 24 MPa, T = 641.77 to 652 K.

1.2 Functions and Inverse Maps

In this first version we concentrated our efforts to produce an accurate library to evaluate properties in regions 1, 2 and 4. These are the regions in wich the water on pressurized water reactors (PWR) exists. THEWASP provides pressure, temperature and enthalpy functions like:

– h, specific enthalpy Kj/m^3
– v, specific volume m^3/Kg
– ρ, density Kg/m^3
– u, specific internal energy Kj/Kg
– s, specific entropy $Kj/(Kg \cdot K)$
– c_p, specific isobaric heat $Kj/(Kg \cdot K)$
– c_v, specific isochoric heat $Kj/(Kg \cdot K)$
– w, sound speed m/s
– x, steam quality %
– μ, dynamic viscosity μPa
– k, thermal conductibility
– Pr, Prandt Number
– st, Superficial tension
– $(\frac{dV}{dT})_p, (\frac{dV}{dP})_t, (\frac{dh}{dT})_p, (\frac{dh}{dP})_t$

and some inverse maps like:

– returns temperature for pressure and enthalpy or entropy
– returns pressure for temperature and enthalpy or entropy

To evaluate derivatives of water and steam properties for regions 1 and 2 we use the approach given by Stücker et al. [9]. In THEWASP we adopted a vector to vector function evaluation to improve speed and suit GPU parallelization. Hence we define three vectors $\overrightarrow{p} = \{p_1, p_2, p_3, \cdots p_{\text{SVECT}}\}$, $\overrightarrow{T} = \{T_1, T_2, T_3, \cdots T_{\text{SVECT}}\}$ and $\overrightarrow{g} = \{g_1, g_2, g_3, \cdots g_{\text{SVECT}}\}$ and function $\overrightarrow{G} = g(\overrightarrow{p}, \overrightarrow{T})$ where g is a property functions. Figure 2 shows the algorithm.

Initialize Library.
Define vector size SVECT.
Allocate memory for host vectors (temperature T, pressure p and $g(p,T)$)
Set p and T values
CALL cuWASP_func(p[],T[], g[],SVECT,**func**)
Where **func** is a association between a ASCII character and a water property $g(p,T)$.
For example, for enthalpy **func** is 'h'
print $\overrightarrow{G}$
Delete memory
END

Fig. 2: vectoreval

1.3 Thermodynamic Coherence

The Gibbs-Duhem equation imposes a general coupling among the partial properties of the components in a thermodynamical system, and is generally the basis of most methods to test their thermodynamic consistency [12] [18].

The total differential of the Gibbs free energy G is given by

$$\mathrm{d}G = \left.\frac{\partial G}{\partial p}\right|_{T,N} \mathrm{d}p + \left.\frac{\partial G}{\partial T}\right|_{p,N} \mathrm{d}T + \sum_{i=1}^{I} \left.\frac{\partial G}{\partial N_i}\right|_{p,T,N_{j\neq i}} \mathrm{d}N_i, \tag{2}$$

Replacing two of the Maxwell relations we have

$$dG = V\,dp - S\,dT + \sum_{i=1}^{I} \mu_i\,dN_i. \tag{3}$$

The potential function is the partial Gibbs free energy, thus

$$G = \sum_{i=1}^{I} \mu_i N_i. \tag{4}$$

The total differential of this expression is given by

$$dG = \sum_{i=1}^{I} \mu_i\,dN_i + \sum_{i=1}^{I} N_i\,d\mu_i. \tag{5}$$

Subtracting the two expressions for the Gibbs free energy total differential gives the Gibbs-Duhem relation

$$\sum_{i=1}^{I} N_i\,d\mu_i = -S\,dT + V\,dp. \tag{6}$$

Considering $I = 2$ for water and steam mixture the chemical potentials across a phase boundary are equal[12]. We assume that temperature and pressure are constant in a neigborhood of the evaluated function. Hence, at constant P (isobaric) and T (isothermal) the activity coefficient becomes

$$\gamma = N_1\,d\mu_1 + N_2\,d\mu_2. \tag{7}$$

Normalizing by total number of moles in the system $N_1 + N_2$ and using the identity $x_1 + x_2 = 1$ we have

$$x_1 \left.\frac{d\ln\gamma_1}{dx_1}\right|_{p,T} = \quad x_2 \left.\frac{d\ln\gamma_2}{dx_2}\right|_{p,T} \tag{8}$$

where x_1, x_2, y_1 and y_2 are properties in a thermodynamical system.

2 GPU Architecture

The GPU used in this work was the GeForce GTX-480, the third generation of the CUDA enabled NVIDIA GPUs. GTX-480 processor is based on Fermi architecture, consists of 4 Graphic Processing Clusters (GPCs), each containing 4 Streaming Multiprocessors (SMs). Each SM has 32 CUDA Cores, four texture units, NVIDIA Poly-morph engine, dedicated caches, register banks and access to DRAM devices via the on-chip memory controller.Like see in 3. The great novelty in this architecture is the migration from SIMD (Single Instruction Multiple Data) to MIMD (Multiple Instruction, Multiple Data), that new concept provides a great improve in parallel computing, so that runs a independent separate kernel function on each of SM unit.

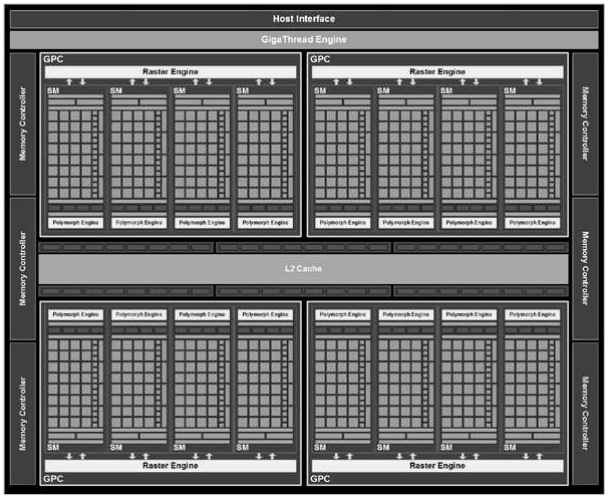

Fig. 3: Simplified GTX-480 GPU graphics processing architecture.

3 CUDA - GPU Programming

CUDA (Compute Unified Device Architecture) is a C-language and SDK compiler that is based on the PathScale C compiler, whose origin refers to OPEN64 project [1] [4]. The GPU global block scheduler manages coarse grained parallelism at the thread block level across the whole chip. When a CUDA kernel is started, information for a grid is sent from the host CPU to the GPU. The work distribution unit reads this information and issues the constituent thread blocks to SMs with available capacity. The work distribution unit issues thread blocks in a round-robin fashion to SMs which have sufficient resources to execute it.

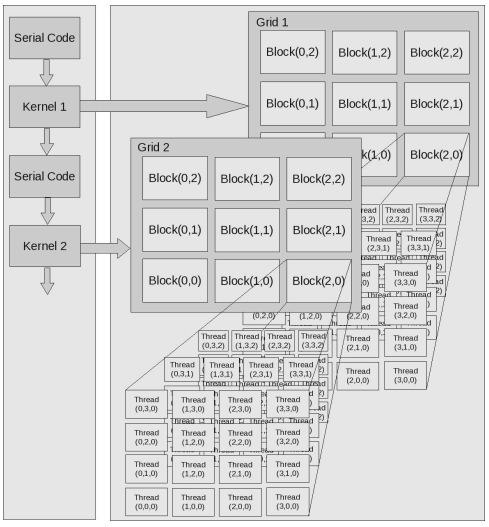

Fig. 4: Flux on CUDA Programming

Figure 4 show simplified flowchart in CUDA program [14]. Some of the factors that are accounted for are the kernel's demand for threads per block, shared memory per block,

registers per thread, thread and block state requirements, and the current availability of those resources in each SM. The end goal of the work distributor is to uniformly distribute threads across the SMs to maximize the parallel execution opportunities.

3.1 Programming Model

In order to simplify the handling of large numbers of threads in Cuda, this language offers the concept of grids and blocks of threads in which our computational domain, up to 1024 threads in x or y dimension, which are divided into sets of blocks called grids, in uni-dimensional or two-dimensional way. Each of these blocks may contain up to 1536 threads arranged in a three-dimensional grid as shown in figure 5.

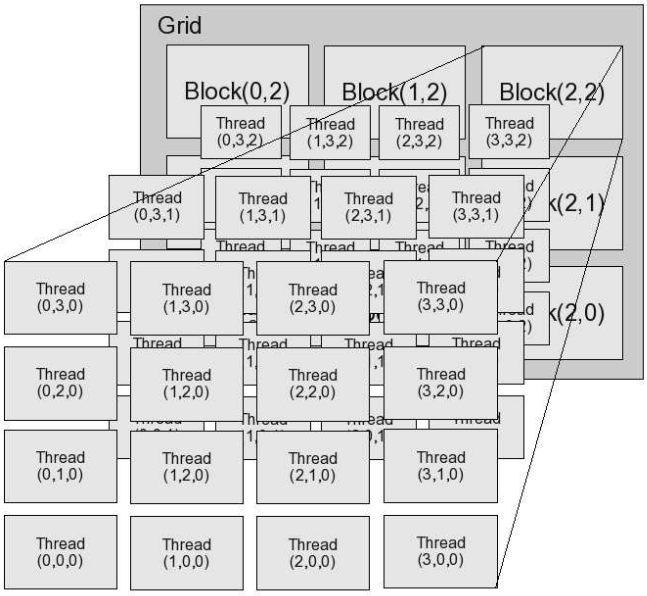

Fig. 5: Grids and Threads Blocks on CUDA

In CUDA 4 was improved the C++ support with virtual addressing, all pointers live in a single address space. This feature enable function pointers, memory allocation using new and delete and recursive function calls. Another improvement in this version is kernel concurrent calls, if hardware available, and so 16 concurrent kernel function can be calling at the same time. A linux 64bits dynamic link library called

`libcuSteam.so`

was build and provides the functions described above.

4 SpeedUp Evaluation

The benchmark test is a loop of functions in the library. Using SVECT = 1024 with fill temperature vector with values ranging from 25 ℃to 350 ℃and do the same with the pressure vector for 0.1 Mpa to 10.1 Mpa.

Allocate memory space for variables
$time = GetTime()$
P {Total pressure = 10.1 MPa}
T {Total temperature T = 350 ℃}
for $p = .1$ to P **do**
  $p \leftarrow p + 10/1024$
  $\overrightarrow{P_i} = t$
end for
for $t = 100$ ℃to T **do**
  $\overrightarrow{T_i} = t$
  $t \leftarrow t + 350/1024$
end for
$\overrightarrow{g} = h(P,T)$, $v()$, $\rho()$, $u()$, $s()$, $c_p()$, $c_v()$, $w()$, $x()$, $\mu()$, $k()$, $Pr()$, $st()$
Print $GetTime() - time$
Delete memory

Fig. 6: Sequential algorithm

The benchmarks run in HP vx8600 workstation with dual 5440 xeons processors and 8 Gigabyte memory and compared with a GPU GTX-480 by NVIDIA, with 1.5 Gbyte of memory. The CPU program is purely sequential and compiled by INTEL C++ compiler 11, based on same library in CPU version. Figure 7 show the parallel CUDA algorithm.

Allocate memory space for variables in CPU
Allocate memory space for variables in GPU
Define a bi-dimensional grid of threads
$time$ = GetTime()
incP = $10/1024$
incT = $350/1024$
$BLOCKSIZEX = BLOCKSIZEY = 64, 32, 16, 8, 4$
Call global function
define a 2D block $(BLOCKSIZEX, BLOCKSIZEY)$
define a 2D grid $(1024/BLOCKSIZEX, 1024/BLOCKSIZEY)$
Call a kernel function $f <<< grid, block >>>$
for each function $f \in$ Library **do**
  $ix = blockIdx.x \cdot blockDim.x + threadIdx.x$
  $iy = blockIdx.y \cdot blockDim.y + threadIdx.y$
  $t = incT \cdot idx + 100$
  $p = incP \cdot idx + 0.1$
  call device function f
  $memory \leftarrow f$
end for
Print $GetTime - time$
Move from device memory do CPU memory
Delete memory from device
Delete memory from CPU

Fig. 7: CUDA algorithm

5 Results

The sequential CPU routine shown in figure 6 performs the test in 189 sec. The parallel GPU routine shown in figure 7 performs the test in 2 sec thus with a 90 times speedup, even when performing a large memory copy (128 Mbyte) from GPU to CPU. Figure 8 shows the Mollier diagram and Figure 9 the temperature-enthalpy diagram both generated by THEWASP library.

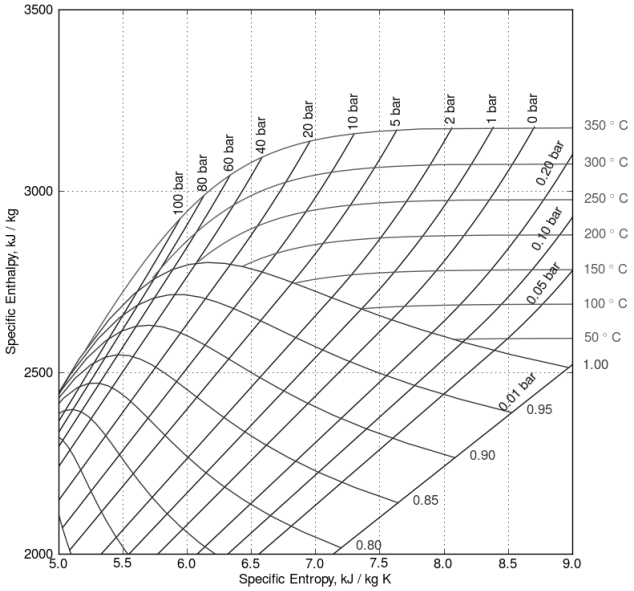

Fig. 8: Mollier Diagram

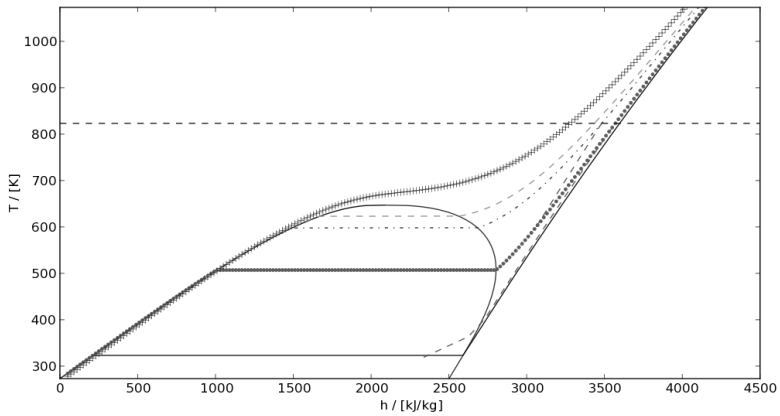

Fig. 9: TH Diagram

6 Conclusion and Future Works

The results here presented show the speed up given by GPU and CUDA programming. Furthermore, the performance of the implementation improves with increased hardware capabilities with no further changes to the code required. With the GTX-480 a maximum speed up of more than 90 times is achieved. Though the speed up in simple kernel function call in GPU has a great potential, several restrictions inherent in planning and also in to the design of the NVIDIA processors reveal great hindrances in the creation of a completely generical library.

This first evaluation of THEWASP encourages us to continue and adapt its use to others programs, like the ones used by reduced scale analysis and thermo-hydraulic power plant evaluation. An application based in this approach can be an *Island Model Genetic Algorithm*, a optimization engine with a great population and a huge number of function calls for water-steam properties. Others applications can be a Lattice-Boltzmann GPU based CFD or a neutronic code with thermo-hydraulic feedback.

References

1. I. Buck. Gpu computing with nvidia cuda. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2007.
2. J. Cooper. Revised release on the iapws industrial formulation 1997 for the thermodynamic properties of water and steam. *The International Association for the Properties of Water and Steam*, pages 1–48, 2007.
3. C.B. Davis. Accuracy based generation of thermodynamic properties for light water in relap5-3d. Technical report, Idaho National Laboratory (INL), 2010.
4. O. Developers. The Open64 web site.
5. W. Gropp, E. Lusk, and A. Skjellum. Using mpi: portable parallel programming with the message passing interface. 1999.
6. A.H. Harvey, A.P. Peskin, and S.A. Klein. Nist/asme steam properties. *NIST Standard Reference Database*, 10, 1996.
7. A. Heimlich, ACA Mol, and C. Pereira. Gpu-based monte carlo simulation in neutron transport and finite differences heat equation evaluation. *Progress in Nuclear Energy*, 2010.
8. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.
9. H.J. Kretzschmar, I. Stöcker, J. Klinger, and A. Dittmann. Calculation of thermodynamic derivatives for water and steam using the new industrial formulation iapws-if97. In *Steam, Water and Hydrothermal Systems: Physics and Chemistry Meeting the Needs of Industry, Proceedings of the 13th International Conference on the Properties of Water and Steam, Eds. PG Hill et al., NRC Press, Ottawa*, 2000.
10. K. Miyagawa and P.G. Hill. A tabular taylor series expansion method for fast calculation of steam properties. *Journal of engineering for gas turbines and power*, 119:485, 1997.
11. KV Moore. Astem: A collection of fortran subroutines to evaluate the 1967 asme equations of state for water/steam and derivatives of these equations. Technical report, Aerojet Nuclear Co., Idaho Falls, Idaho, 1971.
12. M.J. Moran, H.N. Shapiro, D.D. Boettner, and M. Bailey. *Fundamentals of engineering thermodynamics*. Wiley, 2010.
13. C. Nvidia. Compute unified device architecture programming guide. *NVIDIA: Santa Clara, CA*, 83:129, 2007.
14. C. NVIDIA. Programming Guide 2.0, 2008.
15. R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.
16. W. Wagner and A. Pruss. The iapws formulation 1995 for the thermodynamic properties of ordinary water substance for general and scientific use. *Journal of Physical and Chemical Reference Data*, 31(2):387, 1999.
17. M. Waintraub, R. Schirru, and C.M.N.A. Pereira. Multiprocessor modeling of parallel particle swarm optimization applied to nuclear engineering problems. *Progress in Nuclear Energy*, 51(6-7):680–688, 2009.

18. J. Wisniak. The herington test for thermodynamic consistency. *Industrial & engineering chemistry research*, 33(1):177–180, 1994.