Study and Optimization of a Finite Volume's Method Application

José Alves, Rui Brito

Universidade do Minho

Braga, November 2012

Index

Convexion-Diffusion

What? Calculates the heat diffusion of a fluid while it spreads through an area;

How? Uses finite-volumes method;

Why? Represents surface as a mesh, making each cell only dependent of its neighbours;

Convexion-Diffusion

makeFlux Compute the contribution from each edge;

makeResidual Compute the ϕ vector, adding the flux for each cell from each contribution;

LUFactorize Calculate the matrix form of a Gauss elimination;

Test Machines

	compute-511-2@search AMD Opt 6174	compute-611-1@search Xeon X5650	MacBookPro Intel Ivy-Bridge i7
# processors	2	2	1
# cores per processor	12	6	4
hyper-threading	-	yes	yes
clock frequency(GHz)	2.2	2.66	2.3
L1 capacity	128KB	128KB	64KB
L2 capacity	512KB	256KB	256KB
L3 capacity	12MB	12MB	6MB
RAM capacity	64GB	48GB	16GB

Table: Test machines

Test Parameters

- Best of 3 executions within a 5% error margin of each other;
- Test for different number of threads across various systems;
- Single user mode;

Original version

- For each edge:
  - Calculate edge velocity;
  - Calculate flux;
- For each cell:
  - Compute all contributions;
- Compute vector ϕ;
- Compute the matrix form of a Gauss elimination;
- Compute the error;

Challenges

- High number of memory accesses;
- Low operational intensity;
- Deep memory indirection chain;

Optimized version

- Removed redundant loads and calculations;
- Changed some variable definitions to *const*;
- Usage of a recent compiler(SLP);
- Reduced makeResidual's workload from 9.2% to 5.53%;
- Improved computation time from 12.47s to 8.29s;

Counters Used

PAPI_TOT_INS Total instructions;

PAPI_LD_INS Load Instructions;

PAPI_SR_INS Store Instructions;

PAPI_FP_OPS Floating point operations;

PAPI_L1_DCA L1 data cache accesses;

PAPI_L2_DCA L2 data cache accesses;

PAPI comparison

	original version	optimized sequential version
Total instructions	2.517.584	285.551
Load instructions	630.156	86.532
Store instructions	326.459	39.208
FP operations	55.673	44.019
L1 data accesses	1.061.761	153.593
L2 data accesses	22.914	17.467

Table: PAPI comparison

OpenMP Objectives

- Parallelize application;
- Decrease runtime;

Amdahl's Law

$$S_N = \frac{1}{(1-P)+P/N}$$

Parallel Portion	# Cores	Expected Speedup
	1	1
	2	1.0284
	4	1.0433
5.53%	8	1.0508
	12	1.0534
	16	1.0547
	24	1.0560

Table: Theoretical speedups

Achieved Results

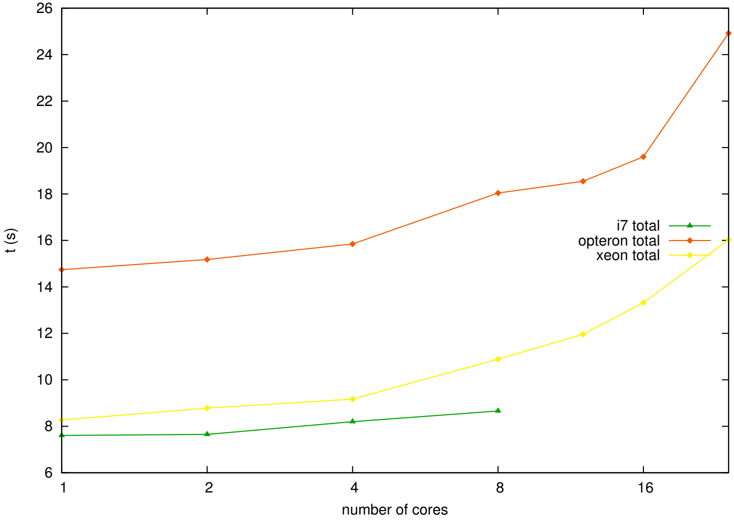

Figure: Total application runtime

Achieved Results

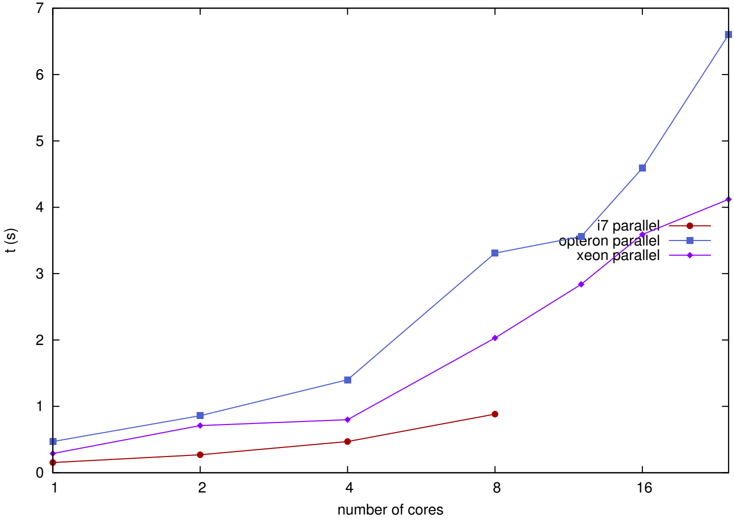

Figure: Parallel section runtime

- GPU version delayed;
- Thorough restructuring of the code;
- FVLib optimizations;
- Improve IO;
- MPI version;

Study and Optimization of a Finite Volume's Method Application

José Alves, Rui Brito

Universidade do Minho

Braga, November 2012