Improving the performance of liquid surfaces modelling in multicore devices

José Ricardo Ribeiro

Supervised by Alberto Proença and José Luís Alves

University of Minho

Master Degree in Computing Engineering

15th December 2015

Overview I

Introduction

- The modelling of liquid surfaces is a process used in simulations and optimization procedures
- Case Study: Brazing components of the Bottom Terminated Component (BTC) type on Printed Circuit Boards (PCB)
  - These BTC components and the respective PCBs go through successive thermal cycles
- The Surface Evolver (SE) is a Computer-Aided Engineering tool to model liquid surfaces
- This work is included in the Bosch HMIExcel Project (2013-2015) as part of the case study 3 of the research line 12

Finite Element Method

- The Finite Element Method (FEM) is a numerical method used to model a problem involving continuous surfaces
- FEM analyses the discrete parts of a surface

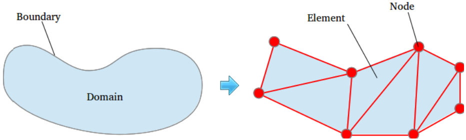


Figure : Spacial discretization of a domain by finite elements

- Each discrete element - the finite element - and the mathematical descriptions of its behaviour contributes to the analysis of the global problem.

Motivation & Goals

- To study the modelling of liquid surfaces with the SE software
- To design and implement a more efficient data structure to deal with vectorization, parallel computing and memory locality
- To improve the sequential version of the SE software
- To implement an efficient parallel version of the application for shared memory environments
- To study how these improvements can be used for a future heterogeneous implementation.

The Surface Evolver

- The Surface Evolver (SE) developed by Ken Brakke at the University of Susquehanna (USA)
- SE is an interactive program to study liquid surfaces, shaped by surface tension and other energies, and subject to various constraints
  - First, the discretization process, when the user writes a model that involves a continuous surface through the analysis of discrete parts of that surface
  - After the model and attributes are defined, then the surface can be evolved
  - Following the iterative process, it outputs the attributes of the evolved surface

Smaller case study

- Initially represented by 10K elements and a memory usage of 1654KB
- Evolves to a surface with 15K elements and a memory usage of 2366KB
- The iterative process is composed by 80 main iterations with computing operations and mesh refinements throughout the iterations.

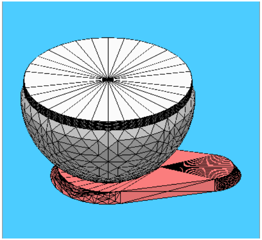


Figure : Initial surface

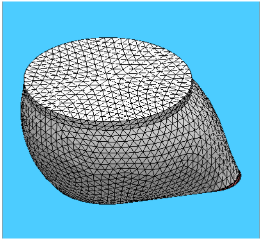


Figure : Final surface

Figure : *Smaller* case surface evolving at the initial and final state

Larger case study

- Initially represented by 500K elements and a memory usage of 314MB
- Evolves to a surface with 1M elements and a memory usage of 978MB
- The iterative process is also composed by 80 main iterations with more intensive computing operations and a more refined mesh throughout the iterations.

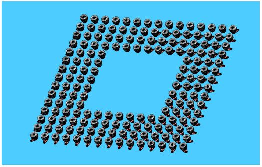


Figure : Initial surface

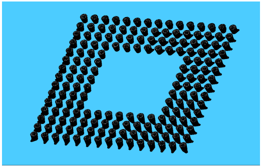


Figure : Final surface

Figure : *Larger* case surface evolving at the initial and final state

Call graph analysis

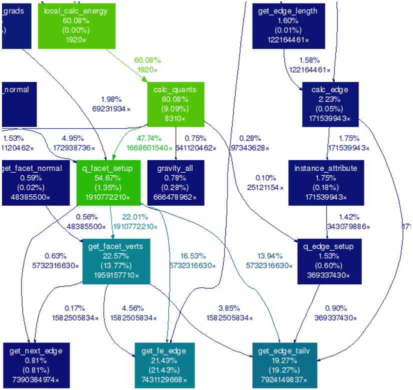


Figure : Call graph identifying the heavier functions

Identifying the heavier functions

Function	*Smaller* case	*Larger* case
recalc	33%	62%
calc_energy	24%	60%
vertex average	14%	7%
calc_force	7%	1%
get_facet_verts, *get_facet_body*	14%	30%

Table : Comparing the impact of the heavier functions in the *smaller* and *larger* case studies

Current parallel implementation analysis

Named quantities

Named quantities are the systematic scheme of calculating global quantities such as area, volume, and surface integrals.

SE parallel implementation

Low-level parallelism implementations using POSIX threads.

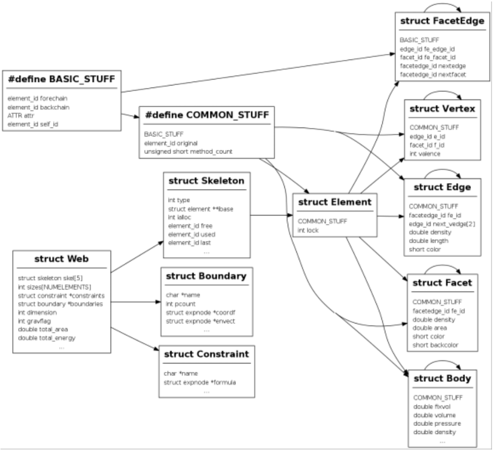


Figure : Main data structure scheme of SE, where the web and the different types of elements and boundaries/constraints are implemented as **linked lists**

Linked lists

- Linked lists can be resized dynamically
- Some issues may decrease the efficiency, both in sequential and parallel computing:
  - *Out-of-order execution*
  - *Hardware prefetching*
  - *Locality of reference*
  - *SIMD*
  - *Heterogeneous environments*

An alternative data structure

Goals of the alternative SE data structure aiming performance:

- *Contiguous memory allocation*
- *Support for Vectorization*
- *Reduce Complexity*
- *Support Parallelism*
- *Avoid False Sharing*

An alternative data structure

This new data structure adds these new maintenance operations:

- *init*: initializes a new structure (allocates space for an initial number of elements)
- *reset (s)*: the structure *s* is reset: (releases the used space and initializes a new structure)
- *set (key, value)*: adds or updates an element *value*, accessible through a *key*, which in this case is always the *id* of the element.
- *unset (key)*: removes an element by its *key* (its *id* in SE).

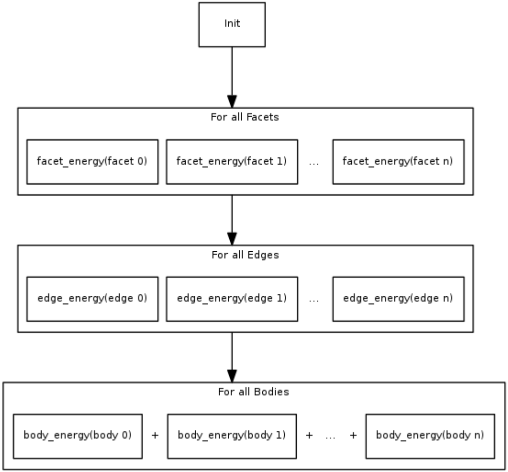
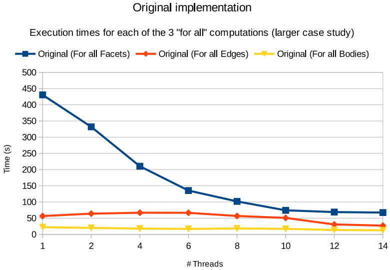
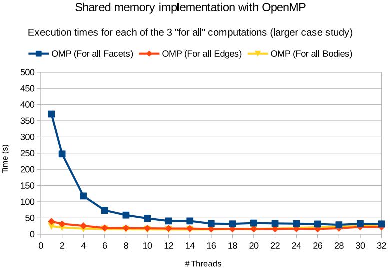


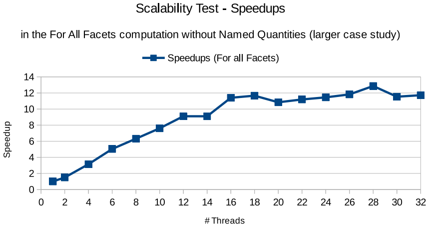
Figure : Main flow of the calc_energy function with computations over all the elements of a specific type

Experimental setup

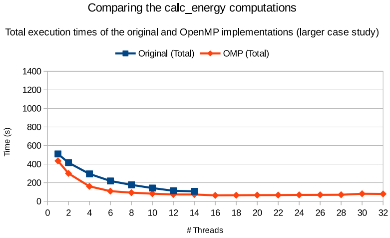
- Testbed: one computer node in a Cluster:
  - Dual 8-core Intel Xeon CPU E5-2650 v2 @ 2.60GHz (with 2-way SMT)
  - Cache per-core:
    - 32KB+32KB L1
    - 256KB L2
  - Cache per-device:
    - 20MB L3
  - Main Memory: 64GB
- Measurement methodology: K-Best approach, where:
  - A minimum of 6 runs and a maximum of 9 runs
  - $K = 3$
  - Tolerance $= 5\%$

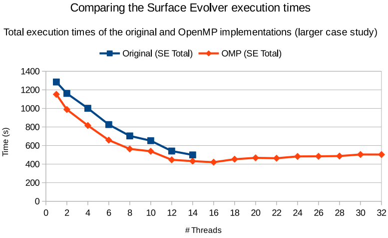

Original implementation
Execution times for each of the 3 "for all" computations (larger case study)
Original (For all Facets)
Original (For all Edges)
Original (For all Bodies)
Time (s)
Threads
0
50
100
150
200
250
300
350
400
450
500
1
2
4
6
8
10
12
14

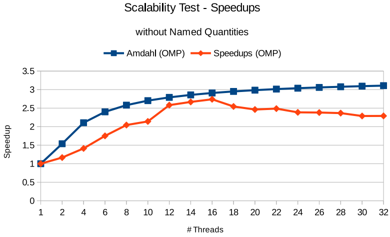

Shared memory implementation with OpenMP
Execution times for each of the 3 "for all" computations (larger case study)
OMP (For all Facets)
OMP (For all Edges)
OMP (For all Bodies)
Time (s)
Threads
0
50
100
150
200
250
300
350
400
450
500
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32

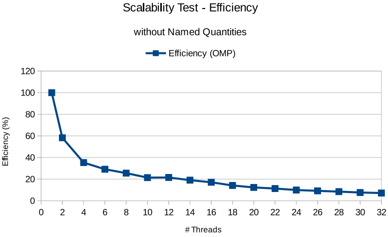

Scalability Test - Speedups
in the For All Facets computation without Named Quantities (larger case study)
Speedups (For all Facets)
Speedup
Threads
0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32

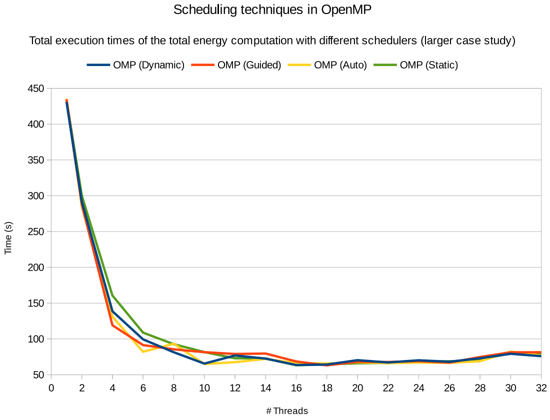
Shared memory implementation

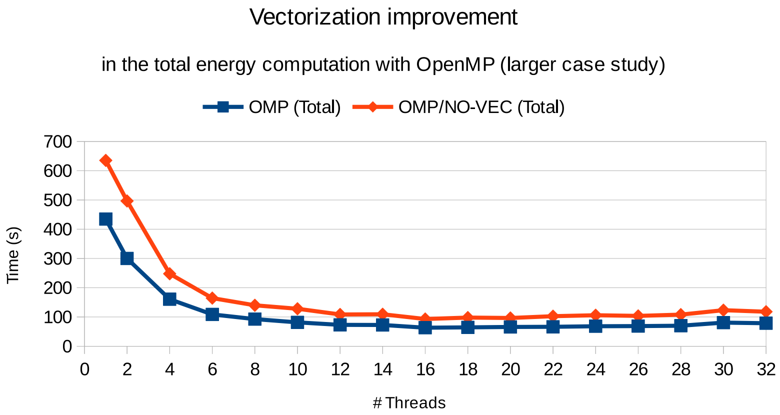









Scheduling techniques in OpenMP
Total execution times of the total energy computation with different schedulers (larger case study)
OMP (Dynamic)
OMP (Guided)
OMP (Auto)
OMP (Static)
Time (s)
450
400
350
300
250
200
150
100
50
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
Threads

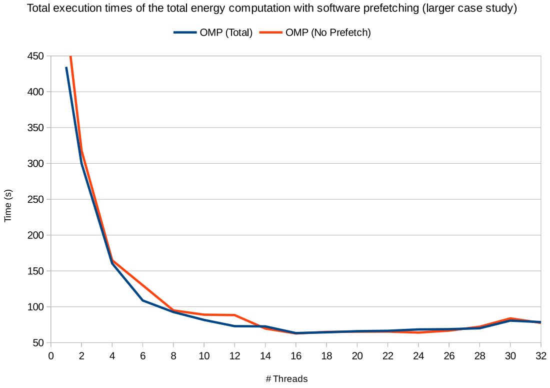

Vectorization improvement
in the total energy computation with OpenMP (larger case study)
OMP (Total)
OMP/NO-VEC (Total)
Time (s)
700
600
500
400
300
200
100
0
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32
Threads

Software Prefetching

Software prefetching and locality optimizations are techniques required for linked lists to overcome the performance gap between processor and memory:

- Benefits of explicit prefetching:
  - *Irregular memory accesses*
  - *Cache locality hint*
  - *Hide latency*
- Negative impacts of software prefetching:
  - *Increased instruction count*
  - *Code structure change*

Software prefetching technique



Conclusions and Future Work

- The alternative data structure improved the SE performance
- More efficient SE parallel implementation

- Remove the critical region in the *For all Edges* computation
- Adapt the alternative data structure to all the functions of the software (including parser and GUI) and if not possible:
  - Improve software prefetching technique
- Explore heterogeneous environments to compute the energy and other quantities

Improving the performance of liquid surfaces modelling in multicore devices

José Ricardo Ribeiro

Supervised by Alberto Proença and José Luís Alves

University of Minho

Master Degree in Computing Engineering

15th December 2015