BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 18 2012, pages 2394–2396
doi:10.1093/bioinformatics/bts454

Structural bioinformatics

Advance Access publication 19, 2012

Cell-Dock: high-performance protein–protein docking

Carles Pons[1,2], Daniel Jiménez-González[3,4,*], Cecilia González-Álvarez[4], Harald Servat[3,4], Daniel Cabrera-Benítez[4], Xavier Aguilar[5] and Juan Fernández-Recio[1,*]

[1]Joint BSC-IRB research programme in Computational Biology, Barcelona Supercomputing Center (BSC), [2]Computational Bioinformatics, National Institute of Bioinformatics (INB), [3]Computer Sciences Department, Barcelona Supercomputing Center (BSC), [4]Computer Architecture Department, Universitat Politècnica de Catalunya, Barcelona 08034, Spain and [5]PDC Center for High Performance Computing, KTH Royal School of Technology, Stockholm SE-100 44, Sweden

Associate Editor: Anna Tramontano

ABSTRACT

Summary: The application of docking to large-scale experiments or the explicit treatment of protein flexibility are part of the new challenges in structural bioinformatics that will require large computer resources and more efficient algorithms. Highly optimized fast Fourier transform (FFT) approaches are broadly used in docking programs but their optimal code implementation leaves hardware acceleration as the only option to significantly reduce the computational cost of these tools. In this work we present Cell-Dock, an FFT-based docking algorithm adapted to the Cell BE processor. We show that Cell-Dock runs faster than FTDock with maximum speedups of above 200×, while achieving results of similar quality.

Availability and implementation: The source code is released under GNU General Public License version 2 and can be downloaded from http://mmb.pcb.ub.es/~cpons/Cell-Dock.

Contact: djimenez@ac.upc.edu or juanf@bsc.es

Supplementary Information: Supplementary data are available at *Bioinformatics* online.

Received on April 26, 2012; revised on June 25, 2012; accepted on July 12, 2012

1 INTRODUCTION

Cell functions rely on intricate networks of protein–protein interactions, which need to be understood at structural and functional level. Given the magnitude of the human interactome, experimental efforts will have to be complemented by computational approaches such as protein–protein docking, which aims to predict the structure of complexes from the free unbound subunits (Ritchie, 2008). Fast Fourier transform (FFT)-based docking approaches revolutionized the field (Katchalski-Katzir *et al*., 1992), but faster computer solutions are required in order to face some of the current docking challenges, some of which have arisen from the recent Critical Assessment of Prediction of Interactions (CAPRI) experiments (Lensink and Wodak, 2010), such as protein flexibility (Grünberg *et al*., 2004; Smith *et al*., 2005), large-scale experiments (Mosca *et al*., 2009; Wass *et al*., 2011) or multi-protein docking (Karaca *et al*., 2011). Given that FFT implementations are highly optimized, it is not likely to speed up this type of calculations by using only algorithm optimization techniques, leaving hardware acceleration as the most feasible option to significantly reduce the computational cost of these tools (Ritchie and Venkatraman, 2010; Sukhwani and Herbordt, 2009). In this work we present Cell-Dock, to our knowledge the first protein docking algorithm implemented on the Cell BE processor (Kahle *et al*., 2005).

*To whom correspondence should be addressed.

2 CELL-DOCK: A VARIABLE GRID RESOLUTION FFT-BASED DOCKING ALGORITHM

2.1 Optimization of FFT parameters

Cell-Dock is a first-stage rigid-body docking program that generates docking candidates performing a global scan of the translational and rotational space of two molecules based on surface complementarity and electrostatics. It implements the Katchalski-Katzir algorithm (Katchalski-Katzir *et al*., 1992), discretizing both receptor and ligand onto 3D grids and using the FFT to speed up the translational search for each given rotation. We based most of our parameters on those of FTDock (Gabb *et al*., 1997). A paramount difference is the fixed value used here for the grid size in number of cells (GRID_SIZE), which is related to the grid cell resolution (CELL_RES) and total span (TOTAL_SPAN) in Angstroms by:

$$\text{CELL\_RES} = \text{TOTAL\_SPAN}/\text{GRID\_SIZE} \quad (1)$$

$$\text{TOTAL\_SPAN} = 1.0 + 2 \cdot \text{radius(rec)} + 2 \cdot \text{radius(lig)} \quad (2)$$

For the sake of efficiency, GRID_SIZE was forced here to be a power of two (that is in practice, 128 or 256; these two versions are identified from now on as CELL-128 and CELL-256, respectively), given that FFT arrays of that length can be usually processed much faster (otherwise FFT performance would depend on the factorization of the array size and the FFT algorithm used). This modification has a direct impact on the CELL_RES parameter, which now is automatically defined by the size of the proteins. As in a realistic docking scenario, we forced CELL_RES to be in the range of 0.7–1.3 Å. Actually, CELL_RES values above 1.3 Å were incompatible with other parameters in our approach, as well as in FTDock, therefore some of the largest cases could not be run in CELL-128.

2.2 Parallel implementation for HPC architectures

We have parallelized Cell-Dock on a PlayStation 3 (PS3) with one Cell BE processor and on a Cell BE Blade (i.e. two Cell BE processors). Parallelization on both architectures was exploited at both task and data level, with the following major features (Fig. 1): (i) the main rotation loop of the docking algorithm was run in parallel on the two Cell BEs of the Cell BE Blade (sequentially in the case of PS3); (ii) 3D-FFT and grid discretization of the mobile protein have been parallelized on the 8 Synergistic Processor Elements (SPEs)—accelerators of a Cell BE (6 SPEs in the case of PS3); (iii) double buffering techniques have been used in order to overlap memory accesses and computation on the SPEs; (iv) data locality policies have been used in order to reduce the amount of data movement between different processors and (v) single instruction multiple data have been used on the PowerPC Processor Element (PPE) running the operating system and SPE codes, as well as dual-threaded processing on the PPE. All those parallelization strategies, together with the variable grid resolution feature of the algorithm, have provided us with a high-performance docking application that can be successfully applied to HPC architectures.

3 SUCCESSFUL PREDICTION OF PROTEIN–PROTEIN COMPLEXES

3.1 Evaluation of docking results

In order to check whether predictive results were similar to other approaches, we tested Cell-Dock on the unbound structures of the protein–protein docking benchmark 2.0 (Supplementary Table S1), formed by 84 cases (Mintseris *et al.*, 2005). For consistency, all success rates were shown here on the 71 cases in which we could run both Cell-Dock versions (see Section 2.1). Given that Cell-Dock used a different grid resolution in each case [CELL_RES; see Equation (1)], we evaluated the consequences of that variable on the quality of the predictive results. First, the method generated a pool of 10 000 docking poses for each case (as in FTDock), which contained near-native (NN) docking solutions (i.e. $RMSD_{LIG} < 10$ Å with respect to the reference) in 89% of the cases when using CELL-256 and in 85% of the cases when using CELL-128. The results are comparable to those of FTDock (Pons *et al.*, 2010) as shown in Supplementary Table S2. We have also assessed the results of scoring the Cell-Dock docking poses by pyDock (Cheng *et al.*, 2007), based on electrostatics, desolvation and van der Waals energy. The scoring by pyDock showed slightly better success rate with CELL-256 (19.7% cases with NN within the top 10 scoring solutions) than with CELL-128 (18.3% top 10 success rate), probably due to grid resolution. Indeed, cases with larger resolution in CELL-128 showed slightly worse results than in CELL-256 (in which almost all of the cases had 0.7 Å resolution; Supplementary Table S1). In any case, these differences were minimal and the general values were similar to those achieved by pyDock when scoring FTDock models (Pons *et al.*, 2010), as shown in Supplementary Table S2.

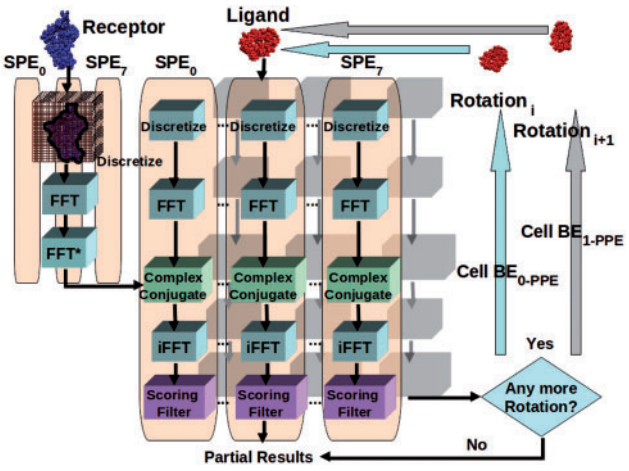

Fig. 1. Scheme of the task partitioning of Cell-Dock on a Cell BE blade with two Cell BEs and 8 SPEs per Cell BE.

3.2 Speedup

The execution times for the 84 benchmark cases with CELL-256 ranged from 45 to 64 min (average time: 48.3 min), on a dual processor Cell BE based blade, which contained two SMT-enabled Cell BE processors at 3.2 GHz with 2 GB DD2.0 XDR RAM (1 GB per processor). In the case of CELL-128, execution times ranged from 3.1 to 4.8 min for the 71 cases in which we could apply this version as above mentioned (average time: 3.73 min). The different performance of these two versions (Supplementary Table S1) can be explained by the increase of data and operations (8× more) in CELL-256 and by the fact that one full plane (128×128) of the grid (128^3) in CELL-128 completely fits in the local memory of a SPE, reducing the communication requirements for the algorithm with respect to CELL-256. For comparison, we ran FTDock on JS21 blades with 8 GB of RAM memory, using only one of the four IBM PowerPC 970MP processors at 2.3 GHz. CELL-256 and CELL-128 achieved speedups of 24.5 and 226.5, respectively, when compared to FTDock at the default 0.7 Å resolution. However, given that the average grid resolution in CELL-128 was around 1.0 Å, it seems more appropriate to compare it with FTDock at 1.0 Å resolution, in which case the speedup achieved was 60.8. We should note that the observed speedups are not due to the difference in clock speed between 2.3 GHz in PPC and 3.2 GHz in Cell BE (which at most would give a theoretical speedup of 1.4 for FTDock on the PPC970) but to the specific features of the Cell BE and the efficient implementation of Cell-Dock.

Cell-Dock was also run on a PlayStation 3 using four of its six SPEs, showing the same predictive results. Executions on a dual-processor Cell BE-based blade (running 8 SPEs each) were 3.8 times faster than on PS3 (one Cell BE with 4 SPEs), which is directly related to the number of processing elements.

4 CONCLUSION

Cell-Dock is a new FFT-based docking implementation for the Cell BE architecture, showing similar prediction rates to FTDock, with much more efficient performance. This implementation can efficiently run on the widely accessible PlayStation 3, opening the door to large-scale experiments that may benefit

from volunteer-based distributed computing. The analysis and performance shown in this work can facilitate the porting of FFT-based applications to other hardware accelerators like Graphics Processing Unit (GPUs).

Funding: This work was supported by the Spanish Ministry of Science and Innovation (BIO2010-22324, TIN2006-27664-E, TIN2007-60625 and CSD2007-00050), Generalitat de Catalunya (2009-SGR-980), the European Commission in the context of the EnCORE project (contract number FP7-248647), the HiPEAC2 Network of Excellence (contract number FP7/IST-217068) and the MareIncognito project under the BSC-IBM collaboration agreement.

Conflict of Interest: none declared.

REFERENCES

Gabb,H.A. *et al.* (1997) Modelling protein docking using shape complementarity, electrostatics and biochemical information. *J. Mol. Biol.*, **272**, 1–3.

Grünberg,R. *et al.* (2004) Complementarity of structure ensembles in protein–protein binding. *Structure*, **12**, 2125–2136.

Kahle,J.A. *et al.* (2005) Introduction to the cell multiprocessor. *IBM J. Res. Dev.*, **49**, 589–604.

Karaca,E. and Bonvin,A.M.J.J. (2011) A multidomain flexible docking approach to deal with large conformational changes in the modeling of biomolecular complexes. *Structure*, **19**, 555–565.

Katchalski-Katzir,E. *et al.* (1992) Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proc. Natl Acad. Sci. USA*, **89**, 2195–2199.

Lensink,M.F. and Wodak,S.J. (2010) Docking and scoring protein interactions: CAPRI 2009. *Proteins*, **78**, 3073–3084.

Mintseris,J. *et al.* (2005) Protein–Protein Docking Benchmark 2.0: an update. *Proteins*, **60**, 214–216.

Mosca,R. *et al.* (2009) Pushing structural information into the yeast interactome by high-throughput protein docking experiments. *PLoS Comput. Biol.*, **5**, e1000490.

Pons,C. *et al.* (2010) Present and future challenges and limitations in protein–protein docking. *Proteins*, **78**, 95–108.

Ritchie,D.W. (2008) Recent progress and future directions in protein–protein docking. *Curr. Protein Pept. Sci.*, **9**, 1–15.

Ritchie,D.W. and Venkatraman,V. (2010) Ultra-fast FFT protein docking on graphics processors. *Bioinformatics*, **26**, 2398–2405.

Smith,G.R. *et al.* (2005) The relationship between the flexibility of proteins and their conformational states on forming protein–protein complexes with an application to protein–protein docking. *J. Mol. Biol.*, **347**, 1077–1101.

Sukhwani,B. and Herbordt,M.C. (2009) GPU acceleration of a production molecular docking code. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*. ACM, New York, NY, pp. 19–27.

Wass,M.N. *et al.* (2011) Towards the prediction of protein interaction partners using physical docking. *Mol. Syst. Biol.*, **7**, 469.