

CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
 SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
 Integrated Infrastructure Initiative

**JRA4 – Final report on production status of initial
 applications**

Deliverable ID: D-JRA4-3

Due date: October 31st, 2005

Actual delivery date: May 17, 2006

Lead contractor for this deliverable: IDRIS – CNRS, France

Project start date : May 1st, 2004

Duration: 4 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	X
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Content

Table of Content.....	2
1. Executive Summary	3
2. Introduction.....	4
3. Optimisation of NCBI-BLAST code	4
4. The porting of the AGMIAL suite	7
5. List of Acronyms and Abbreviations	12

1. Executive Summary

This document is one of the two PM18 deliverables of the Joint Research Activity in Life Sciences. It contains a final report on the deployment and operation of the two genomic projects initially supported by IDRIS, and a report on the BCS activity in the area of protein interactions since PM13 (date at which BCS joined the DEISA project).

This document is publicly available.

2. Introduction

This deliverable constitutes the final report on the optimisation and operation of the two genomic applications initially selected by IDRIS for migration to the DEISA platform:

- Identification of new human mitochondrial proteins (INSERM project). The purpose of this project is the high-throughput identification of new human mitochondrial proteins by “in silico” comparative genomics. The identification of these nuclear mitochondrial genes would allow a better understanding of mitochondrial diseases.
- Large scale microbial genome re-annotation (INRA project). Re-annotation all known prokaryotic genomes (194) using the AGMIAL platform developed at the MIG INRA laboratory in France, and provide to the scientific community unified data mining bioinformatics tools to explore this huge amount of data.

3. Optimization of the NCBI-BLAST code for the INSERM project: final report (Olivier Glorieux, Denis Raux, Anne Fouilloux, Isabel Dupays, IDRIS – CNRS)

3.1 Description of the project:

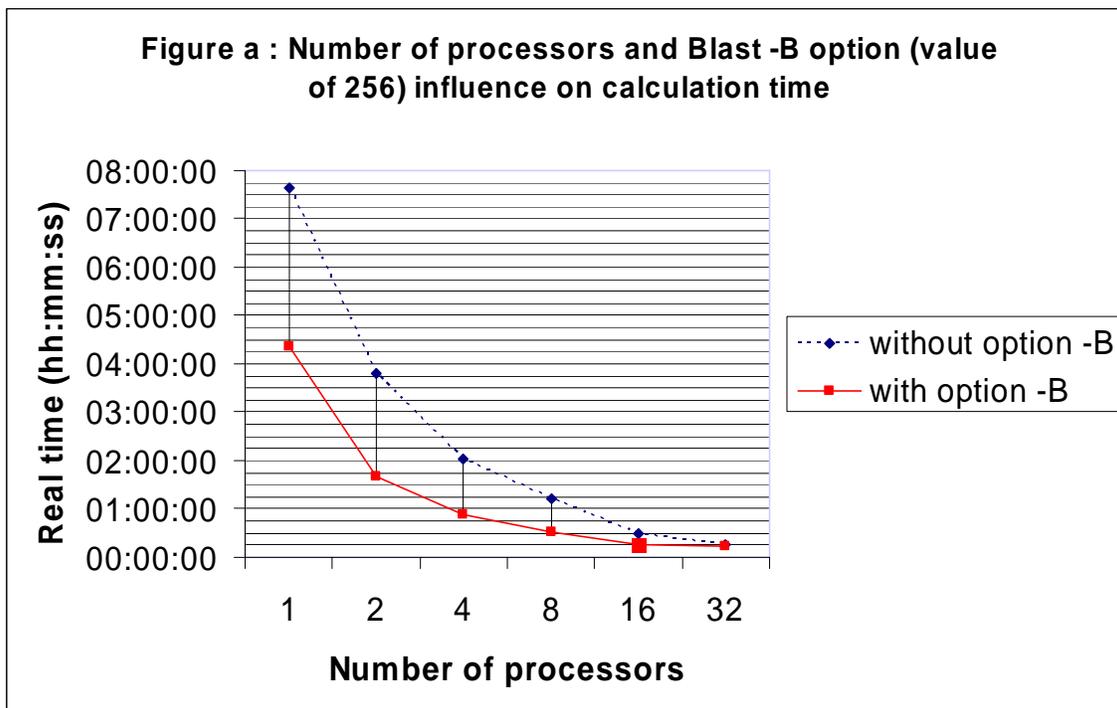
An INSERM team had to conduct a very large BLAST (tblastn): 600 000 (about 0.2 Gbases) prokaryotic proteins against the human genome (about 3 Gbases). The estimation time for that BLAST on their best machine, a Sunfire 280R Bi-processeur/4Go, was of 540 days (for 400,000 proteins) which was prohibitive. Details about the potential scientific impact of this simulation were extensively provided in the previous deliverables dealing with this project (see D-JRA4-2a).

3.2 Optimization

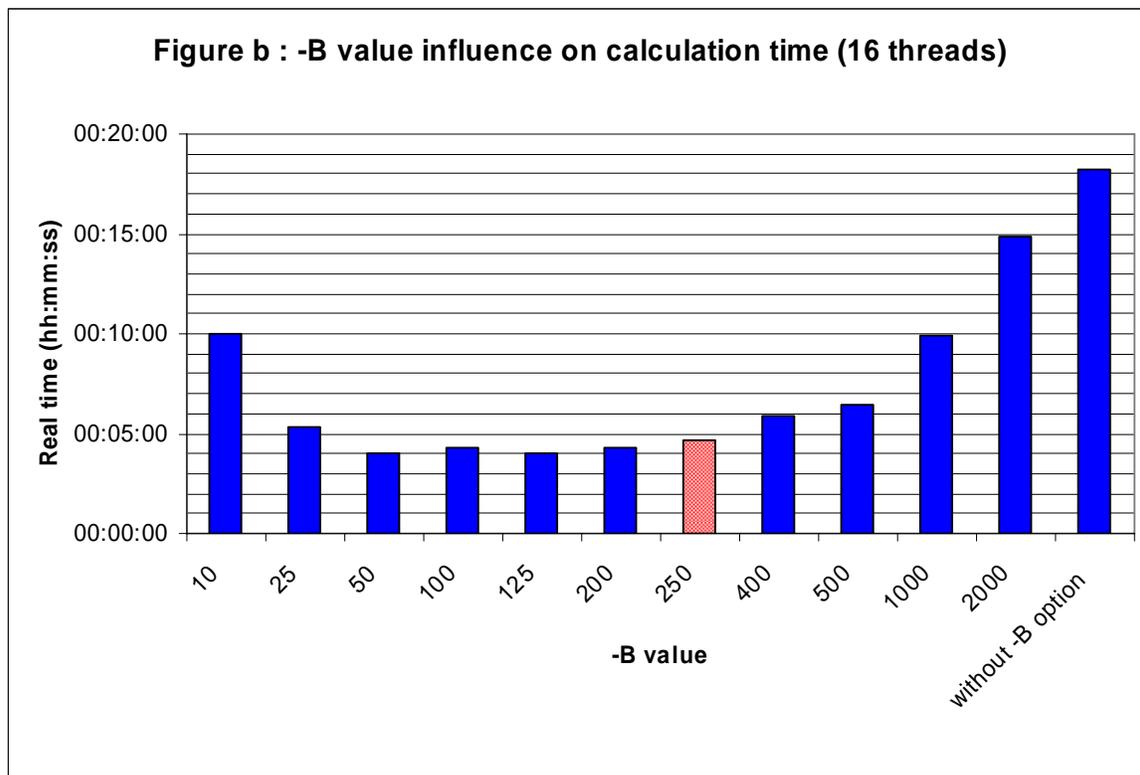
NCBI's BLAST was ported to the IDRIS IBM SP4 supercomputer (Zahir), installed the needed databanks (human genome). The code was fine tuned to run in a shared memory node, parallelized with threads. Two aspects needed to be explored: the level of parallelization, and the way input and output data should be managed.

The work done for the optimization of this code was described in all detail in a previous deliverable (D-JRA4-2a). The production runs were performed during the summer 2005. We will summarize here the main features of the code finally used for the production runs.

The Figure A shows the speedup obtained using the parallel version of Blast combined with it's -B option (detailed in the Figure B).



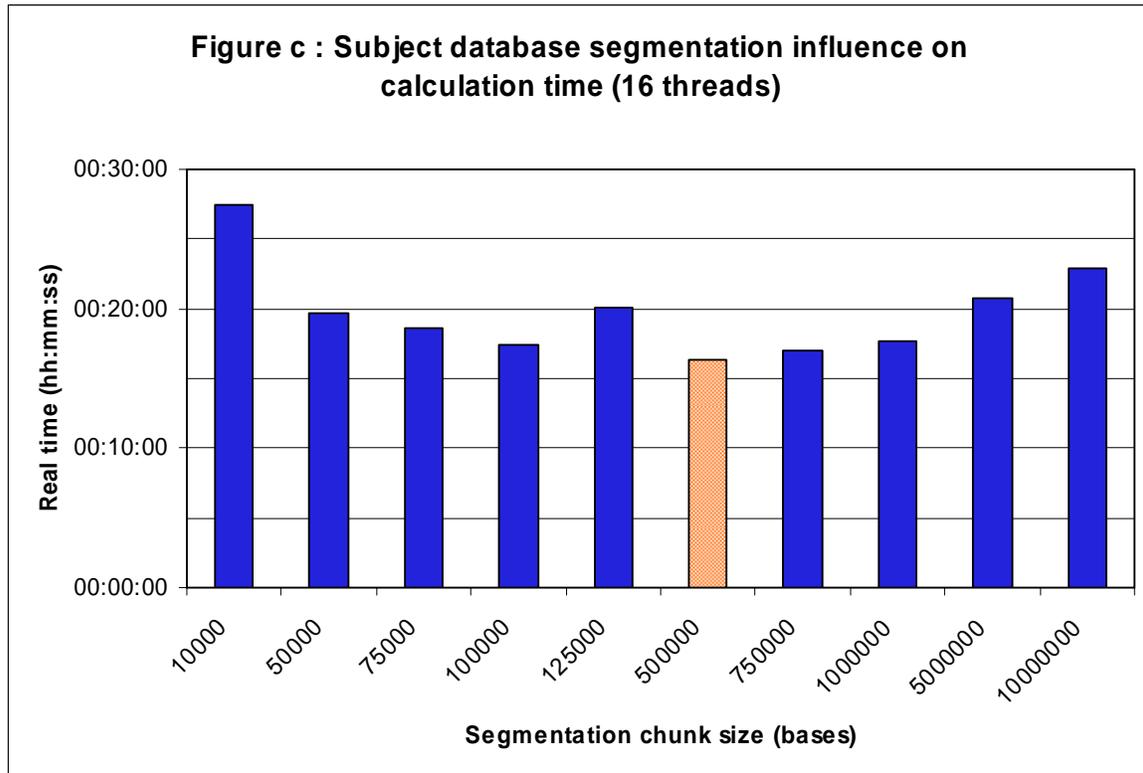
All the production runs were performed on 16 processors. After a scanning and testing of the available option of the threaded BLAST, the -B option (query concatenation) was chosen so as to accelerate the calculation, lengthening the size of the resulting global query to an optimal value (Figure B).



For practical reasons (number of files to manage), the production runs were performed with a -B value of 256. Two flaws in the implementation of that -B option were noticed:

the limit of the number of queries to be concatenated (255) and the non-standard output which made unreadable the results. We improved this situation, and NCBI validated those modifications, available in the next release.

The threaded blast code needs the subject database to be segmented in sequences in order to split the workload to each processor. We had to determine the best value for that segmentation (Figure C).



Production runs were made with a segmentation chunk size value of 500000 bases. The typical final tblastn command is:

```
blastall -p tblastn -i AllBact001.faa -d Hg17 -a 16 -B 256 -o AllBact001.out
```

The calculation time, for the whole project, was reduced from the original **540 days to 8 days** on the IBM SP4 platform (code running on 16 threads).

3.3 Future Directions

The scientific impact of the production runs is being currently assessed by the INSERM users.

With the enormous and increasing size of biologic databases, the need of powerful machines like Zahir and the other platforms of the DEISA Grid grows, not only to mine the database itself but also to mine the outputs. The increasing Teraflops of DEISA's

heterogeneous grid of supercomputers will soon be essential to many other ambitious and innovative projects

4. The porting of the AGMIAL project to the IBM SP4 platform: final report (Olivier Glorieux, Denis Raux, Anne Fouilloux, Isabel Dupays, IDRIS – CNRS)

4.1 - Description of the project :

The MIG team of INRA-Jouy developed a genome annotation platform called AGMIAL whose purpose is to automatically annotate microbial genomes. In order to obtain annotated genomes with controlled quality and traceability, the MIG team wanted to reannotate the whole known genome database (about 200 microbial genomes at the time) using AGMIAL. In the INRA laboratory such a task required a prohibitive calculation time: between 500 and 1000 days. This is why this proposal was adopted as a DEISA project. The initial purpose was to port AGMIAL on the IBM SP4 (Zahir) platform and make the resulting annotated genomes database accessible to INRA.

4.2 - Description of the AGMIAL annotation platform:

AGMIAL is basically a Java-based wrapper of a set of stand alone diverse public applications; it consists of two main managers one for nucleic entries and the other for the proteic entries deduced from the nucleic data. Each manager is a collection of independent applications that will scan the biologic sequence and derive annotations that will be integrated and stored in a relational database.

4.3 - The actual portage:

It quickly appeared that a complete porting of the AGMIAL annotation platform was unrealistic: a large number of its components were simply not adapted to run on a supercomputer, and a deep modification of its structure was required. This was clearly beyond the scope of the support that DEISA could provide to this scientific team. It was however decided to port all the annotation applications wrapped in AGMIAL, and we identified three main time consuming applications on which the emphasis should be set.

- SHOW, an 'in house' application of the MIG team dedicated to locate the genes that will be translated in proteins and passed to the protein manager.
- BLAST, essential alignment software and
- HMMER, specialized in detecting patterns and families.

4.4 - HMMER Optimization:

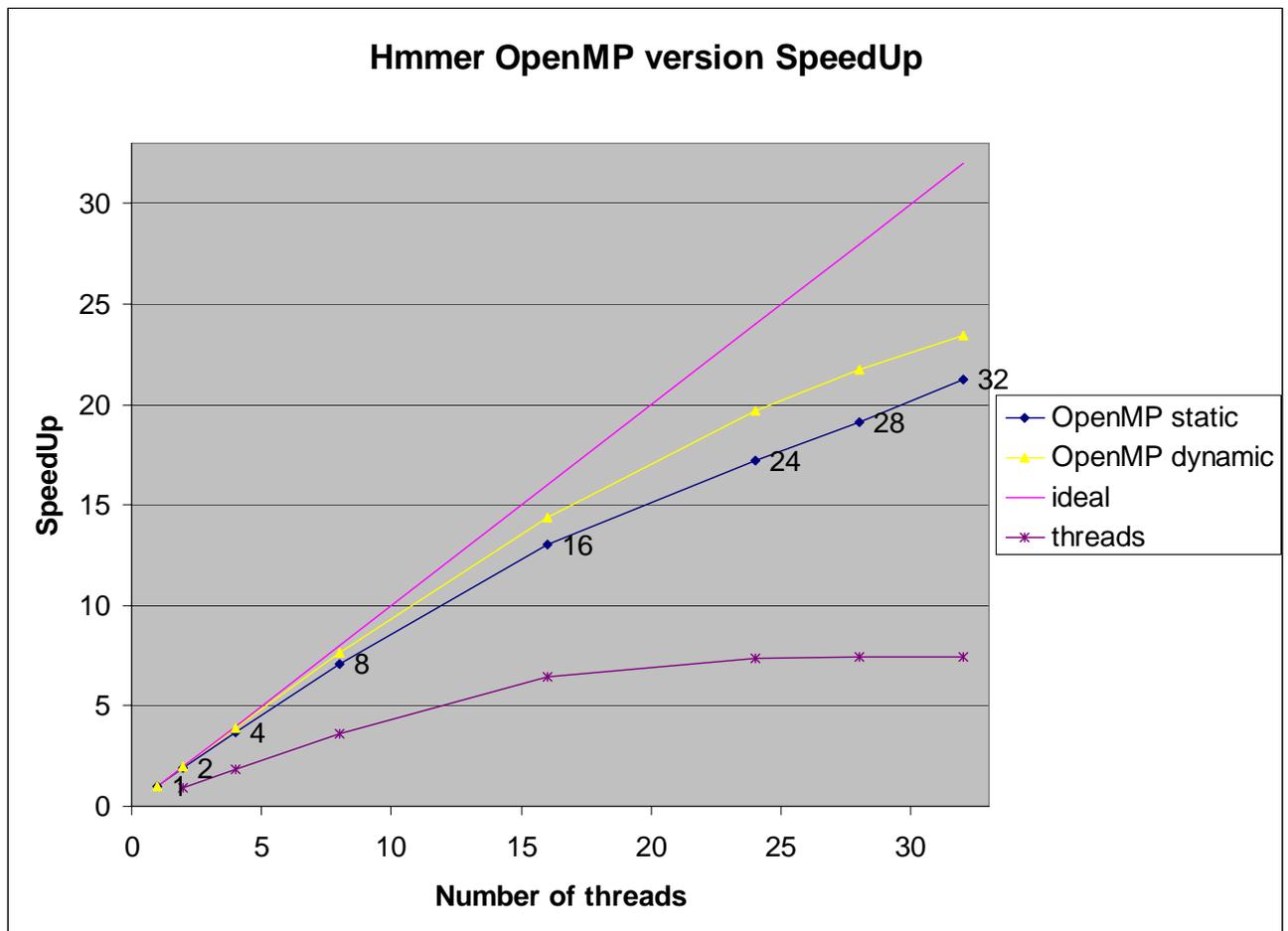
4.4.1 --mem Option

Since the summer 2005, it was decided to focus on HMMER, and its “hmmpfam” functionality (the one used in AGMIAL). After porting HMMER on Zahir, it was realized that the input/output part was not optimal, because hmmpfam was implemented for small memory configuration. This is why a new option was coded: '--mem' that allows storing the whole hmmer database once, into memory. (See annex 1)

This configuration reduces the execution time by a factor of two.

4.4.2 - OpenMP parallelization

A first approach of parallelization has then been accomplished. We implemented an Open MP version of hmmpfam. Calculation time was very substantially reduced using this new version '#ifdef HMMER_OMP' :



The blue curve represents the OpenMP version speedup when the repartition of the work load between the processors is static. (Each processor will receive an equal and final portion of the hmm database)

The yellow curve represents the OpenMP version speed up when the repartition of the work load between the processors is dynamic; each processor receives a new portion of the hmm database, of a given size (10) when it has finished processing it. Here we notice that the dynamic configuration has better speed up than the static one which tell us that the time to process an hmm pattern is not constant and depends maybe on its size.

The garnet-red curve is the already existing threaded version of HMMER that we can compare to our new OpenMP version. The OpenMP implementation, dynamic or static has better speed up than the threaded version, which is also without the `-mem` option.

In AGMIAL, it is the original version of hmmpfam without threads that is run. Therefore, the execution time would be reduced by a factor of 29 using our version running on 16 processors (which is a good compromise value: great acceleration and speedup close to ideal)

These modifications were submitted to the author of Hmmer who is still looking at them and assessing the best way to incorporate them into the generic application.

4.4.3 - Conclusion :

The optimization performed on HMMER has an intrinsic value in itself, independent of the insertion of HMMER in the AMIAL suite.

The initial project has come to an end because of the lack of investment of the INRA scientific teams. IDRIS and DEISA cannot provide the enormous development support that the full migration of AGMIAL would require. AGMIAL is not suitable for large scale calculations in its current implementation. See the discussion of this issue in D-JRA4-2a.

We are still working on the parallelization of Hmmer, and plan to try an MPI version because we believe it is a useful tool for the general effort in bioinformatics.

4.5 - Annex 1: hmmpfam code modifications: --mem option

We allocate an array of hmm structures, which will contain all the hmms of the database and then we call HmmStore to do the actual storage.

```
/* IDRIS 22/08/05 allocation of the hmmTab array and calling of the  
HmmStore function */
```

```
struct plan7_s *hmmTab = NULL ;
```

```
if(do_mem){
```

```
hmmTab = calloc(ORIGINAL_HMM_TAB_SIZE, sizeof(struct plan7_s));
```

```

    /* IDRIS 05/09/05 if the calloc failed (maybe because there is not
    enough memory available) */
    if(hmmTab == NULL) {
        Die("Failed to allocate the plan7_s array in which we store the
        hmmer database. ORIGINAL_HMM_TAB_SIZE : %d",ORIGINAL_HMM_TAB_SIZE );
    }

    nhmm = HmmStore(hmmTab, hmmfp);

}

```

The HmmStore function a while loop go through all the hmm structures and stores them into the array previously allocated

```

/* Function: HmmStore()
* Date:      05/07/05
* Author : Olivier Glorieux
*
* Purpose:  Stores every Hmm of the Base into the memory
*
* Args:     hmmTab - an array where to store hmms
*           hmmfp  - open HMM file (and at start of file)
*
* Returns:  int nbHmm the number of Hmm stored
*/

/* IDRIS 05/07/05 creation of this function to store hmms */
static int HmmStore(struct plan7_s hmmTab[], HMMFILE *hmmfp)
{
    int nbHmm = 0;
    struct plan7_s    *hmm;

    while (HMMFileRead(hmmfp, &hmm)) {

        //If we reached the end of the original array, reallocation of a new
        array two time the size
        if(nbHmm > ORIGINAL_HMM_TAB_SIZE){
            Die("the number of hmms in the database : %d, is greater than the
            ORIGINAL_HMM_TAB_SIZE. \n Please #define at compiling time a greater
            ORIGINAL_HMM_TAB_SIZE", nbHmm);
        }
        //storing into the dedicated hmm array
        hmmTab[nbHmm] = *hmm ;
        //FreePlan7(hmm);
        nbHmm++;
        //fprintf(stderr, "nbHmm = %d\r", nbHmm);
    }
    return nbHmm ;
}

```

4.6 - Annex 2 : hmmpfam code modifications : OpenMP parallelization

It is the loop that goes through each sequence to be scanned that is going to be parallelized: first we set the variables that will be private, specific to each sequence.

```

/* IDRIS 29/08/05 OMP directive that specifies what variables are

```

```
private */
#pragma omp parallel private(hmm, mx, thresh_loc)
```

The initial ‘while’ loop has been replaced by a ‘for’ loop necessary to use OpenMP directives

```

#pragma omp for
for(nhmm=0; nhmm < *ret_nhmm ; nhmm++) {

    hmm = &hmmTab[nhmm] ;

#ifdef HMMER_OMP

    inside_loop_serial(hmm, hmmfile, seq, sqinfo,
                      &thresh_loc, do_xnu, do_forward, do_null2,
                      ghit, dhit, nhmm, dsq, mx);

#else

    inside_loop_serial(hmm, hmmfile, seq, sqinfo,
                      thresh, do_xnu, do_forward, do_null2,
                      ghit, dhit, nhmm, dsq, mx);

#endif

```

We use the critical directive in a part of the code where each value must be processed sequentially, here it is for the sorting of the scores (results)

```

/* IDRIS 29/08/05 OMP directive that forces threads to process one
after the other for that part of the code */
#pragma omp critical
sc = PostprocessSignificantHit(ghit, dhit,
                              tr, hmm, dsq, sqinfo->len,
                              sqinfo->name, NULL, NULL, /* won't
need acc or desc even if we have 'em */
                              do_forward, sc,
                              do_null2,
                              thresh,
                              TRUE); /* TRUE -> hmmpfam mode */

```

6. List of Acronyms and Abbreviations

AGMIAL	Annotation de Génomes Microbiens d'Intérêt Agro-Alimentaire
BLAST	Basic Logical Alignment Research Tool
IRISA	Institut de Recherche en Informatique et Systèmes Aléatoires
INRA	Institut National de la Recherche Agronomique
INSERM	Institut National de la Santé et de la Recherche Médicale
NCBI	National Centre for Biotechnology Information
