



# BGBLAST: A BLAST Grid Implementation with Database Self- Updating and Adaptive Replication

*Gabriele A. Trombetti, Ivan Merelli,  
Alessandro Orro, Luciano Milanese*

*CNR-ITB, CILEA, INFN, CNRS*



**dkfz.**





- **BGBlast and Biological Databases Handling in Grid**
- Objectives
  - Provide Grid support for BLAST, greatly reducing user wait times for large BLAST runs
  - Provide main BLAST Biological Databases in the Grid Environment
  - Optimize Grid storage and transfer costs
  - Keep databases updated
  - Handle versioning
    - co-existence of multiple versions
    - handle naming, name clashes
    - store versions in a cost-efficient manner



- BLAST Basic Local Alignment Search Tool
  - Compares unknown “Query” sequences to “known” genomic or proteomic reference sequences
    - Uses a reference flat-file database (BRD Blast Reference Database)
    - BRDs are very large, order of magnitude of 500MB – 5GB
  - BLAST = Variation of exhaustive dynamic-programming Smith-Waterman algorithm
    - Not exhaustive anymore, reduces sensitivity slightly (*still acceptable in most cases*)
    - Improves speed 10x - 100x, ***but this can be still insufficient***



- Solution to insufficient BLAST speed:
  - Faster alternatives to BLAST
  - Cluster execution
  - Grid execution



- Solution to insufficient BLAST speed:
  - Faster alternatives to BLAST
    - Solutions available e.g.
      - MegaBLAST
      - BLAT
      - PatternHunter
    - Lower sensitivity or closed source, might or might not be acceptable
    - BLAST reliability well established and cannot be object of discussion (research/publication purposes)



- Solution to insufficient BLAST speed:
  - Cluster execution
    - E.g.
      - Y. Qi, F. Lin: **Parallelisation of the blast algorithm**, *Cell Mol Biol Lett.* 2005;10(2):281–5
      - D.R. Mathog: **Parallel BLAST on split databases**, *Bioinformatics* 2003;19(14):1865–6
      - A. Darling, L. Carey, and W. Feng: **The Design, Implementation, and Evaluation of mpiBLAST**, *4th International Conference on Linux Clusters* June 2003; San Jose, CA
    - Usually regarded as reliable, however:
      - Initial cost for the cluster can be high
      - Uneven workloads -> Cluster underused



- Solution to insufficient BLAST speed:
  - Grid execution
    - Existing solutions, e.g.:
      - F. Konishi, and Y. Shiroto, and R. Umetsu, and A. Konagaya: **Scalable BLAST Service in OBIGrid Environment**, *Genome Informatics* 2003;14:535–6
      - I. Merelli, L. Milanese: **High performance implementation of BLAST using GRID technology**, *Proceedings BITS* 2005: p.59
    - Unsolved problems:
      - Defining and enforcing policy for BRD allocation on Grid SEs
      - Keeping BRDs updated
      - Keep older version of BRDs available if possible



- BGBlast (Group 3 - BLAST execution in Grid)
  - BGBlast advantages:
    - BLAST parallel execution on Grid
    - Automatic update of BRDs on Grid SE nodes
    - Adaptive replication of BRDs over Grid SE nodes
    - Provides version regression of BRDs



- BGBlast parts
  - GridBlast core
  - Adaptive Replication Manager (ARM)
  - Automatic Database Updater (ADU)
  - Database Version Regression engine (DVR)



- BGBlast parts
  - GridBlast core
    - Merelli-Milanesi's GridBlast provides the core functionality
      - I. Merelli, L. Milanesi: **High performance implementation of BLAST using GRID technology**, *Proceedings BITS 2005*: p.59
    - Factor J parallelization of large BLAST execution by splitting input into J even subsets. Launch on Grid of J independent, parallel jobs.
    - Rate limiter
    - Monitoring of J launched jobs, resubmission on failure
    - Fetching of J results back, merging results
    - Monitoring of queue/execution times (---> used in ARM engine)



- BGBlast parts
  - Adaptive Replication Manager (ARM)
    - Motivation:
      - BRDs are large files usually in the range 500MB – 5GB, and are needed near (=LAN) the execution location of BLAST jobs
      - It is not possible to know usage of each BRD in advance
        - This constraints the number of CE which can be chosen for a job:
        - Few replicas? --> Grid not fully exploited. Slow on large BLAST runs.
        - Many replicas? --> Storage costs too high
    - Solution: ARM optimizes the number of replicas of BRDs on Grid SE nodes dynamically and adaptively



- BGBlast parts
  - Adaptive Replication Manager (ARM) – Methods
    - ARM adaptively replicates more the BRDs which have recently been used more often, and less (or 1 time only) the others
    - D-days moving average computation (usually  $D=10$ ) of the usage (CPU hours) for each BRD
    - When raising the number of replicas: also computes the most advantageous SE for replica addition (avl.disk space,# of near WN)
    - When lowering the number of replicas: also computes the least advantageous of currently existing replicas, best for removal



- BGBlast parts
  - Adaptive Replication Manager (ARM) – Algorithm details
    - ARM iterative algorithm minimizes the cost formula:
      - $\text{cost} = \text{storage cost} + \text{user wait time cost}$
    - Formula is computed for N (current), N+1 and N-1 replicas.
      - Best result is chosen --> number of replicas is adapted
    - Cost for N+1 and N-1 is simulated. Inverse proportionality of queue and execution times to number of available CPUs is assumed (simplifying assumption). No. of available CPUs for BGBlast can be computed exactly, from the location of the replicas.



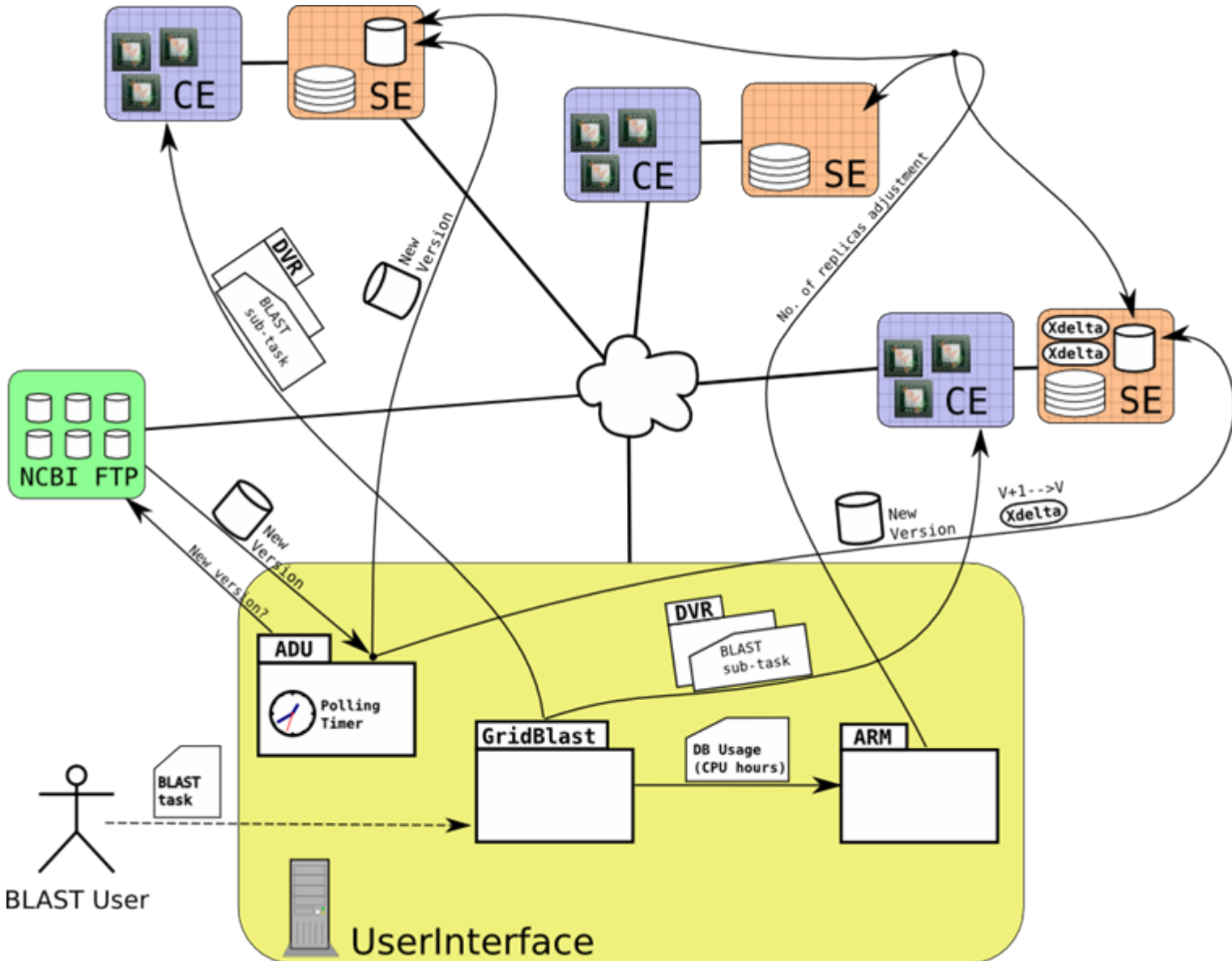
- BGBlast parts
  - Automatic Database Updater (ADU)
    - ADU engine constantly monitors FTP sites for new versions of the BRDs
    - When a new version “V+1” is detected:
      - Version V+1 of the BRD is uploaded over all replicas
      - ADU computes an Xdelta patch to regress the NEW version to the OLD version (V+1-->V). Patch = small file representing differences
        - Xdelta patch “V+1-->V” is uploaded on the Grid
        - Xdelta patch “V+1-->V” is much smaller than the BRD (V), and is not replicated. Storage costs are bearable



- BGBlast parts
  - Database Version Regression engine (DVR)
    - A *date* option can be specified when launching BGBlast, together with the BRD name. Date in ISO standard YYYYMMDD
    - DVR engine agent **moves** with the Blast job. On the WN, DVR regresses the BRD applying Xdelta patches previously generated
    - Xdelta patches regress the BRD by one version at a time. They are applied in sequence until the requested date is reached.
    - Xdelta patches are not replicated ==> remotely downloaded. Slow operation, but uncommon also.



# BGBlast Summary View

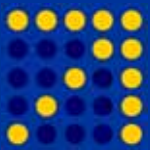




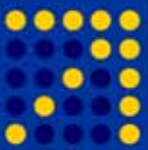
- Preliminary testing and tuning
  - Size of databases
    - Max=5.3GB (nt.gz); **Avg=691MB**;  $\sigma / \bar{x} = 1.59$
  - Replication speed (bandwidth testing)
    - Avg=1.91MB/sec;  $\sigma = 1.02$ ;  $\sigma / \bar{x} = 0.53$
  - Replication balance (hours/day n-replicas equilibrium)
    - Up to now we have tuned the parameters so that:
    - 1GB file will get 1 additional replica for each 10 CPU-hours/day of average computation (moving average over 10 days).
    - 2GB files get half that number of replicas, etc.
    - We will perform more exact cost computations/optimization



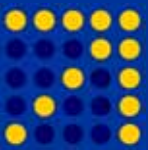
- Biologist needs to blast 50,000 sequences against NR version dated 2005-06-15
  - Blastp CPU time against NR: 25 seconds per sequence  
Total: 15 CPU days (Xeon class CPU)
  - This can be performed on the Grid with a 100+ speedup



- How to do that with BGBlast:
  - From the commandline on an UI w/ bgblast installed
  - `bgblast.pl -p blastp -i input.seq -d nr --version 2005-06-15`
  - this will take care of
    - splitting the input.seq reasonably
    - launching J parallel jobs
    - regressing the database on the WN to 2005-06-15
    - waiting for the J partial results
    - merging the partial results in a single results file



- The following Blast Reference Databases are currently maintained:
  - BLAST databases:
    - nr (NCBI),
    - nt (NCBI),
    - pdbaa (NCBI),
    - UCSC\_human\_chrs (UCSC),
    - human\_genomic (NCBI),
    - refseq\_protein (NCBI),
    - refseq\_rna (NCBI),
    - refseq\_genomic (NCBI),
    - ecoli (NCBI),
    - yeast (NCBI),
    - uniprot (UNIPROT),
    - est\_human (NCBI),
    - est\_mouse (NCBI)



- Also maintained with the ADU engine:
  - InterPro databases
    - PROSITE Patterns (Hofmann, K. et al. 1999),
    - PROSITE profile (Hofmann, K. et al. 1999),
    - PRINTS (Attwood, T. K. et al. 2000),
    - Pfam (Bateman, A. et al. 2000),
    - PRODOM (Corpet, F. et al. 1999),
    - SMART (Schultz, J. et al. 2000),
    - TIGRFAMs (Haft, D.H. et al. 2001),
    - PIRSF,
    - PANTHER,
    - SUPERFAMILY



- This work was supported by the Italian FIRB-MIUR project “Italian Laboratory for Bioinformatics Technologies – LITBIO” [15] and by the European Specific Support Action BioinfoGRID [16] and EGEE [17] projects
- Special thanks to:
  - Ivan Merelli
  - Alessandro Orro
  - Luciano Milanese

