Reference UPC-IO Implementation

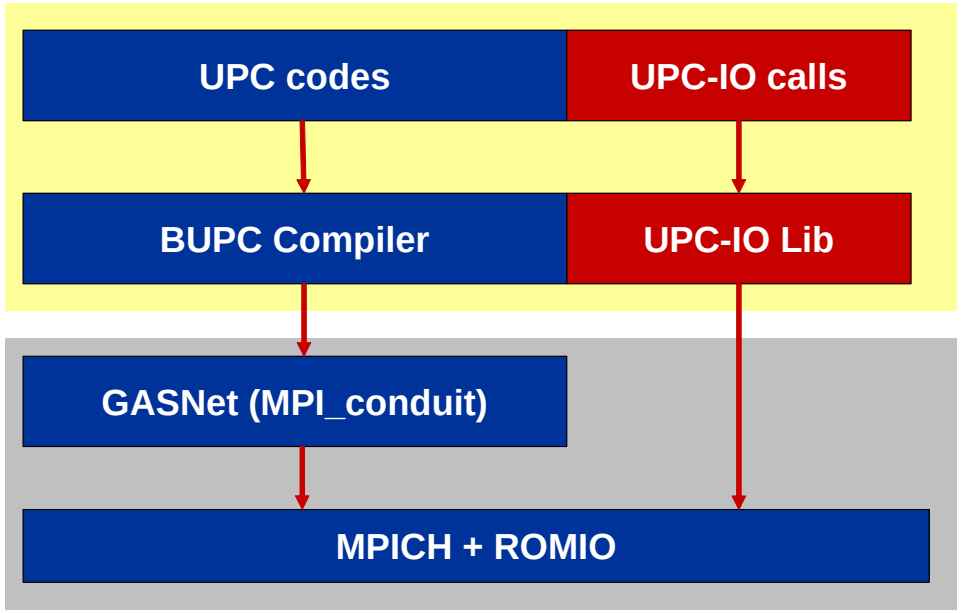
Status and Report

Reference UPC-IO Status

- **The first release is available online since May, 2005**
- **Internal CVS**
  - **Keep tracking of all the versions**
  - **Change logs**
    - **Beta 1 – Nov, 2004**
      - » Initial version. Concept verified and basic functionality implemented.
    - **Beta 2 – Jan, 2005**
      - » Mapped each UPC-IO function to the most optimized MPI-IO optimization levels.
    - **Beta 3 – March, 2005**
      - » Further optimized the read/write functions and clear naming space
    - **Release (Beta 4) – May, 2005**
      - » Major bug fixed. Passed the test suites and internal cases.

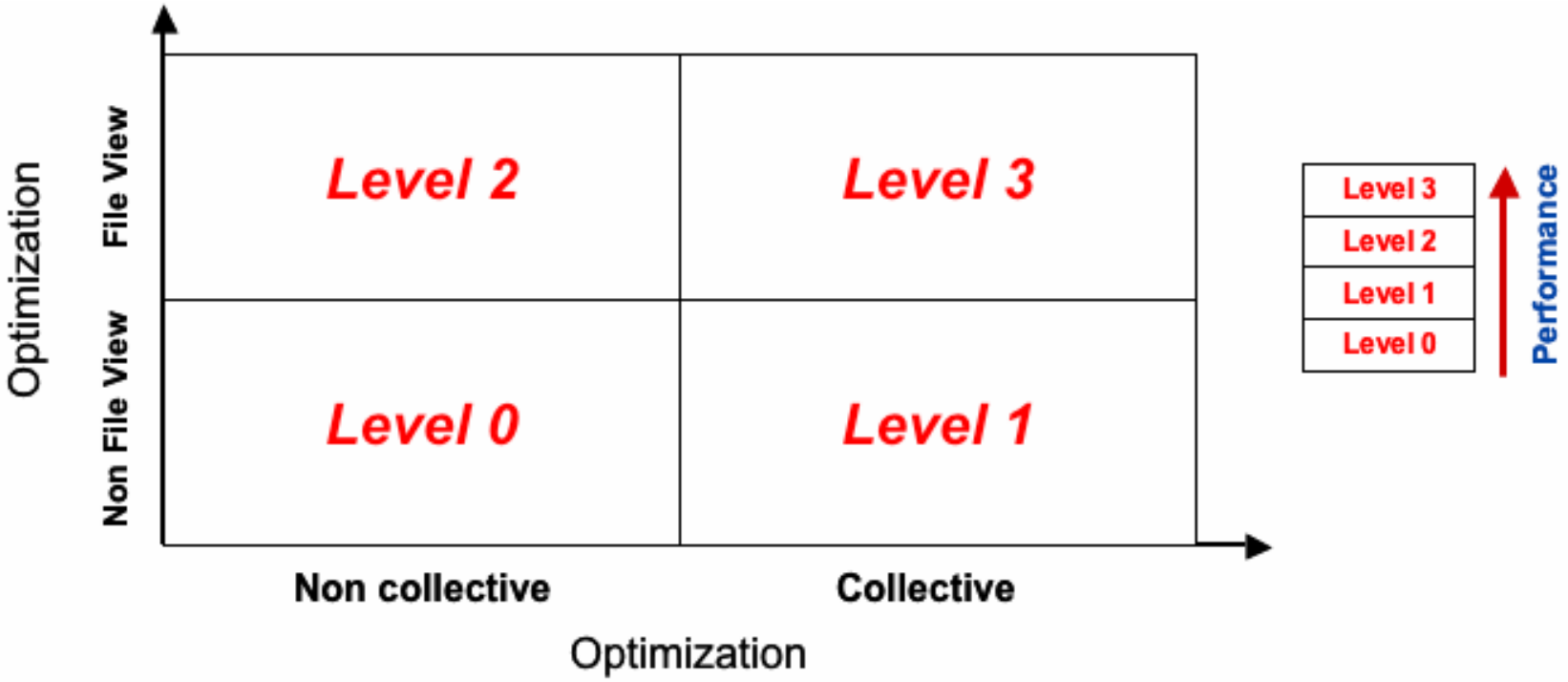
The Reference Implementation

- Based on MPI-IO
- Implements full UPC-IO library functions
- Needs Berkeley UPC compiler (supports MPI)
- Spec v1.2 compliance
- Virtually portable to any platform that provides MPI-IO support

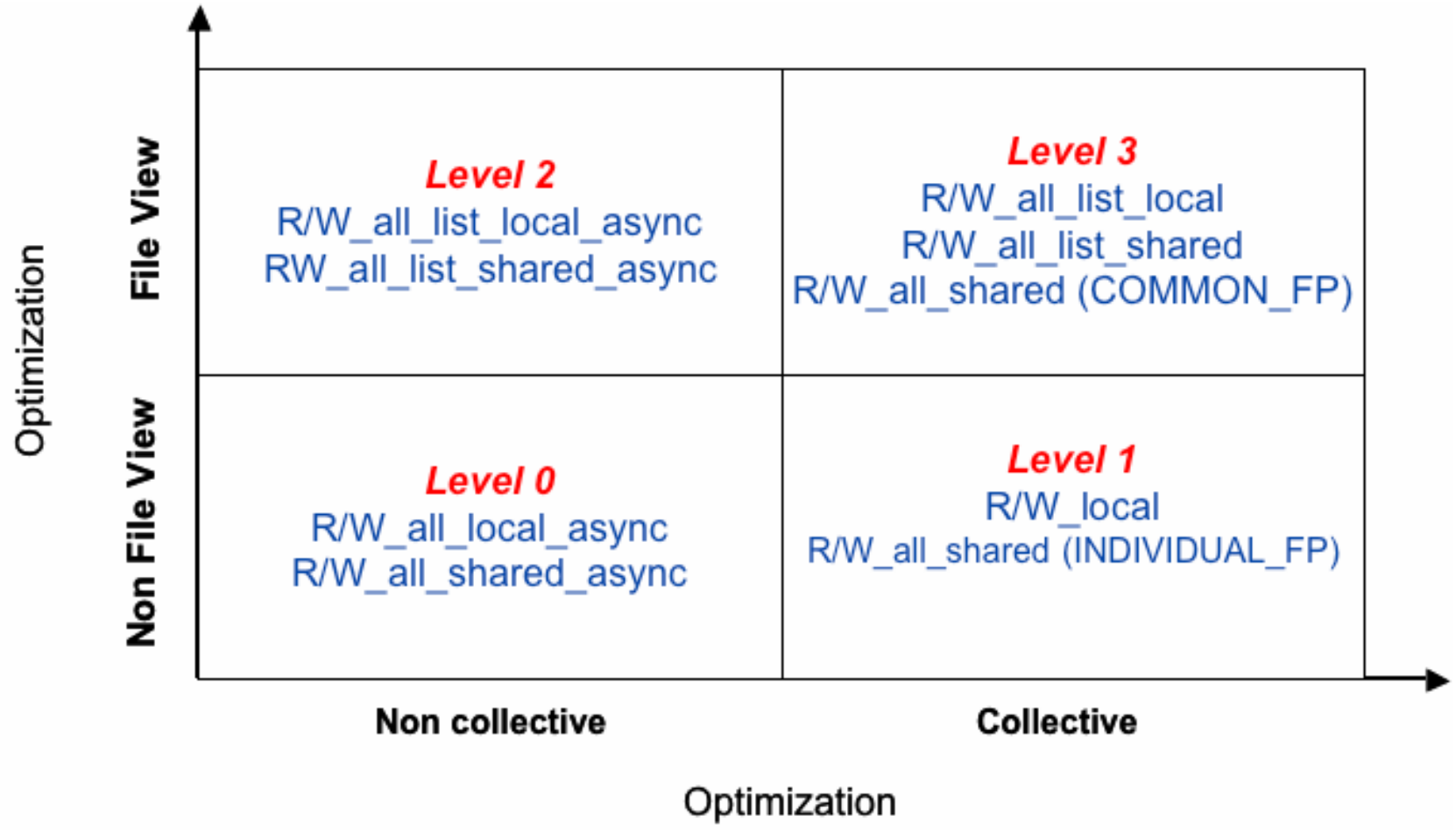


MPI-IO Optimization Levels

- MPI-IO provides 2 axis of optimizations
- Higher level usually delivers better performance



UPC-IO Optimizations



Mapping UPC-IO functions onto MPI-IO optimization levels

Testing

- The Formal UPC-IO testing – Black box
  - Testing strategy, strictly following the latest UPC spec v1.2, feature-by-feature
  - Testing suites, guided by the strategy
- Internal testing cases – White box
  - Trying to cover erroneous paths
  - Corner cases, e.g. misaligned shared array
- Comprehensive test case from Berkeley group (Dan)
  - Help to identify the problem from users' point of view

Benchmarking

- Synthetic Benchmark
  - Read/Write from/to local buffer
    - with and without fsync function calls
  - Read/Write from/to shared buffer
    - both private and common file points
    - infinite and finite blocksize
  - Stress testing number of opened files
- Real application: BTIO
  - Based on NPB BT application

Future Plans

- A portable high performance mid-layer
  - Similar to *ADIO* (Abstract-Device Interface for I/O) for the MPI-IO, which directly talk to the File system to enable the high performance
  - Reference implementation to support PVFS
- More I/O benchmark
  - e.g. SSCA#3 with File IO