**Toward Tunable Parallel File Operations on Lustre-Based Systems**

**v0.1 (as of 3 Jan 2014)**

**Overview**

* The issue
  + Petascale getting common, users running at larger scales, number of users sharing the system growing
  + Easy to overwhelm even the most robust file system
  + I/O practices not keeping pace with scale
  + Big gap between user readiness and need
  + Technologies exist, but not well understood
  + Still treating file system as one big hard disk
  + Still reading and writing as if one user, one task
  + Performance bottlenecks at individual job level
  + But also users stepping on each other
  + A single user can bring down the system
* The need
  + Need for more robust approaches – genuine, efficient, effective parallel i/o
  + Need for research to understand performance issues and optimization
  + Need for education
    - Advocacy and visibility – why this is important
    - Perspective – how parallel file systems like Lustre work, and the implications
    - Avoiding basic mistakes
    - Value of (and options for) transitioning to effective, efficient, parallel i/o
    - Tutorials
    - Best practices
* The initiative
  + TACC engaged in initiatives on all three fronts
  + Inform and affect each other
* This paper is
  + report of early results, emphasis on the tools and research
* Acknowledge on-going work
  + The neural net guys. How our approach differs

**Enabling Tuning**

* Fundamentals of Lustre in a paragraph -- especially stripe count and stripe size
* Fundamentals of MPI-IO, ROMIO, and higher level libraries in a paragraph
  + Explain that ROMIO implementation includes the interface to the file system and can be "Lustre aware" in the sense of using hints and tunable parameters
* Two parameters always available: stripe count and stripe size
  + Explain
  + Plausible defaults
  + But "optimal" depends on many things, including what you're going to do with file
  + Tunable for writes; fixed property of file for reads
* Given parallel file ops based on MPI-IO (this includes libraries like HDF5, NetCDF, Adios)...
  + Can perform collective or independent parallel i/o
  + If collective, number of "writers" (informal term for aggregators) is a third important parameter (meaningless for independent i/o
* Explain MPI\_INFO object

**t3pio**

We’ve developed t3pio – a user friendly tuning tool for setting three key parameters important in parallel write ops based on MPI-IO (including HDF5 and NetCDF)

Fortran and C/C++

Link to the library, call the function, then use the results

Specify as much or as little as you want

Plausible defaults -- explain what they are

Typical call(s) in one language; the other is similar

Does this by setting MPI\_INFO object

So you actually have to use it

Note -- can set striping on the directory before execution

But can't set writers (and writers is a meaningless concept for independent)

Just turning it on is a big help!

Explain why this is true, even though there are default stripe parameters on Lustre that are supposed to be "pretty good"

Available for download at...

**Preliminary Results**

* Very hard problem
  + Huge parameter space: performance depends on many independent variables
    - "Givens" include...
      * Architecture. Not only Stampede vs Lonestar, but also $WORK vs $SCRATCH
      * xxxxx
      * xxxxx
    - "Knobs" include...
      * xxxxx
      * xxxxx
      * xxxxx
    - Other factors include...
      * Choice of libraries
      * Choice of collective vs independent parallel i/o
      * Other implementation details
      * xxxxxx
  + Performance varies enormously
    - Load imposed by other users
    - "Phases of the moon" (e.g. a temperamental OST)
  + Optimal choices depend on purpose
    - If you're going to write a file once then read it a thousand times in a largemem shared memory environment, you may not want to choose striping parameters to optimize the write operation!
  + Competing factors: variability argues for repetition, but huge parameter space argues for limited number of experiments of short duration. Like looking through a drinking straw.
  + A lot of folks working on this
    - Neural nets
* Began using t3pio to conduct exploratory studies -- see what patterns emerged (or at least suggested themselves)
  + For various file sizes, hold two knobs fixed and vary the third
  + Two codes (C++ and Fortran) independently
    - Hyperslab writes of simple arrays
    - Collective and independent
  + Typical results
* Sometimes counter-intuitive
  + Collective isn't always faster
  + Smaller files seem to benefit from larger stripe sizes, and size greater than 2MB of no value
  + The "chasm" anecdote: unfortunate choices of stripe count can affect performance catastrophically. These are implausible choices (who in the world would go out of their way to set stripe count equal to a large prime?), but they help inform and even validate emerging model
  + Can't just "time the write"
    - Which write? task 0?
    - A lot of time can be spent in close
    - But not seeing much variation across processes
    - Any reasonable timing metric will do, just don't call it throughput (it's dimensionless, but we time the whole thing and call it effective throughput)

**An Emerging Performance Model**

* Assumptions
  + MPI-IO (including
  + File size (either total file size, or file size per MPI task) is determined by scientific needs and thus can be viewed as fixed.
  + Very small files outside scope of this work -- looking at mid-sized -- low GB to low TB
  + Focused on write for now
* Gut feel -- subject to validation
  + Performance varies from experiment to experiment, but optimal knob settings do not. In other words, given a fixed set of conditions (architecture, tasks, file size), and a fixed intended file op (e.g. 160 tasks needing to write certain data to a 50GB HDF5 file), the optimal choice of the three knobs does not vary from experiment to experiment.
  + On a given file system, the optimal choice of each of the three knobs is function of files size, but is largely independent of the values of the other two knobs. For example, optimal stripe count is a simple function of file size, but is largely independent of stripe count and writers.
  + Variations across file systems can be characterized as parameters. For example, if knob a is a linear or quadratic function of file size, we can determine appropriate parameters defining that function for the $SCRATCH file system on Stampede.
  + There are plausible defaults. In fact, rules of thumb already suggesting themselves:
    - writers = stripe count = nodes
    - stripe size = 1-2MB
    - Larger stripe counts and stripe sizes do not help and may hurt performance

**Future Work**

* We're now validating the model
  + e.g. is optimal stripe count independent of other two?
  + For each of three file sizes...
    - For each of three values of stripe size
      * For each of three plausible values of writers
        + Determine optimal stripe count
  + Explore collective vs independent
* Once valid, tuning parameters for various architectures
* Plausible parameters for generic architectures
* Embed model in t3pio

---------------------------------------------------------

**Notes to colleagues:**

**I think we get a paper out of each phase:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Tools** | **Research** | **Education** |
| **Phase 1** | **t3pio: One-Step Tuning**   * User/programmer calls the tuning function, specifying as much or as little as desired * Tuning function includes plausible but not optimal defaults | **Exploration**   * What factors affect i/o performance? * What patterns emerge? | **Visibility and Advocacy**   * Why this is important * Basics of Lustre * General advice: common mistakes and good citizenship * Transitioning to genuine parallel i/o |
| **Phase 2** | **t3pio: mechanism for research**   * Parameter studies to refine and validate performance model * Xxxxx * xxxxx | **Modeling**   * Develop, refine, and validate model for setting tuning parameters | **Tutorials**   * Basics of MPI-IO, HDF5, NetCDF |
| **Phase 3** | **t3pio: One-Step Optimization**   * Tuning function includes performance optimization model | **Optimization**   * Performance model reflects tuning parameters for meaningful inputs and architectures | **Best Practices**   * Tuning for optimal performance |