# BG/L System Software Challenges

- Automatic SIMDization for double-hummer
- Working with the limited per-node memory footprint
- Extracting the last 30% of I/O bandwidth

# Automatic SIMDization for double-hummer

- Compiler improving greatly with each release
- Problem is difficult with alignment restrictions, stride=1 restriction, and dependence of register pairing
- Can quantify with benchmarks comparing –q440 vs. –q440d
- Collaborate by publishing guides and best practices
  - http://www-1.ibm.com/support/docview.wss?uid=swg27007511
  - Could BG/L consortium collect best practices from members?
  - Could BG/L consortium share hand-tuned libraries?

## Working with the limited per-node memory footprint

- Release 3 supports racks with 1GB per node
- Memory is a substantial cost of a rack
- Would a mix of 1GB and 512MB nodes help?
- Are there useful coding techniques that can be shared to ease the problem?
- Can the “pooled node storage” option be useful?
  - Can it be totally transparent?
  - If not, how much awareness of paging does an application need?
  - How general purpose would it need to be?

## Extracting the last 30% of I/O bandwidth

- Currently we show the Linux can exceed 100MB/sec direct via NFS
- Addition of CNK and protocol limits performance to 70MB/sec aggregate per ION with NFS
- Initial challenge is to tune a file system capable of > 70MB/sec/ION
  - Network must be tuned to eliminated bottlenecks
  - Fileservers must be capable of delivering > 70MB/sec
  - Can we share our techniques for tuning?
- With a fast fileserver how do we extract the last 30MB/sec (read)?
  - Need to analyze more closely where performance is lost
  - Compare with other similar clients (e.g. ppc32 mac)
  - Compare with other filesystems (e.g. GPFS can reach 90MB/sec/ION)
  - Use the 2nd cpu or tree interrupts?