1. After some experimentation, I've discovered that the lowest specifications with which my simulator can reliably hit above 95% is with at least 8192 bytes allocated to the L1 cache, full associativity, between a 32-128 byte block size (the optimal for the gcc trace is 128 bytes, and the optimal for the ls traces are 32), and a least recently used replacement algorithm. In general, full associativity and the LRU algorithm always appears to return the highest hit rate, and the block size varies greatly on a case to case basis.
2. I've also discovered that the lowest specs with which my simulator can reliably hit above 99% is with at least 32768 bytes allocated to the L1 cache, full associativity, either 256 or 512 bytes for the block size (the optimal for the gcc trace is 512, and 256 for the ls traces), and a least recently used replacement algorithm. Once again, full associativity appears to generally be the correct call, as with LRU, while the block size will vary greatly depending on the cache sizes, the associativity, and the trace file you're using.
3. After some experimentation, I've discovered that the best compromise appears to be to give the L1 16384 bytes of memory and the L2 131072 bytes with both of them set to full associativity, LRU, and and a block size of 1024 bytes. However, the way that we're testing the L2 cache is weighted against it. In the implementation as we have it, the L2 cache is only checked if we already know that the requested address was not found inside of the L1 cache, and is not checked if the address does exist in the L1 cache. Although this makes the most sense for a real world implementation, it means that, for a test like this, it artificially makes the L2 cache appear to be much less efficient than it really is. Due to the nature of the situation (the L2 cache always contains the full contents of the L1 cache), every time that the L2 cache is not checked, it's guaranteed to have actually had the address cached. As such, its hit rate appears to be far below what it actually is. If we modify the code so that the L2 cache is always checked, even if the address is already found in the L1 cache, the test that I previously described jumps up to a 95% hit rate for the L1 cache, and a 99% hit rate for the L2.