부하분산 메타데이터 카탈로그 서비스의 성능 분석

Performance Analysis of Load Balanced Metadata Catalog Service

안선일, 이세훈, 김남규, 황순욱
한국과학기술정보연구원

Sunil Ahn, Sehoon Lee, Namgyu Kim, Soonwook Hwang
KISTI

요약

메타데이터 카탈로그 서비스는 그리드 상에 저장된 파일들에 대한 메타데이터에 대한 접근을 제공한다. 본 논문에서는 자주 사용되는 메타데이터 카탈로그 서비스 중의 하나인 AMGA의 성능을 분석하였다. DB에 대한 직접 접근에 비해 약 700%의 오버헤드가 있었으며, 이 중 가장 큰 오버헤드는 GSI/SSL을 처리하는 오버헤드로 약 350%를 차지하였다. 본 논문에서는 이 오버헤드를 줄일 수 있는 방법 중의 하나로 부하분산 기법을 제안하고, 이 기법의 성능을 측정하였다. LAN 환경의 경우 AMGA 서버의 증가에 따라 성능 향상이 가능함을 확인하였다. 그리드 상의 작업이 직접 AMGA에 접근하는 경우 데이터베이스에 직접 접근하는 것만큼의 성능 향상을 얻기 위해서는 AMGA 내부에 DB 연결 풀링 기법과 부하분산 기법을 활용하는 것이 필요함을 확인하였다.

Abstract

AMGA is a metadata catalogue service which offers access to metadata for files stored on the Grid. We evaluated the performance of AMGA and analyzed overhead in the current AMGA implementation. It had 700% poor throughput (read/insert per second) compared with the direct DB access. The biggest overhead was in managing GSI/SSL Connections, degrading throughput about 350%. We alsomeasured the throughput of load-balanced AMGA services, and it showed linear throughput improvement when we increased the number of AMGA server. In addition, we measured the throughputin the WAN environment, and it showed the number of the maximum connections that an AMGA server can handle is the most critical factor in the throughput.

I. 서 론

데이터 그리드에서 파일은 대개 여러 분산된 스토리지 사이트에서 관리된다. 사용자가 관심있는 파일을 찾기 위해서는 파일에 관련된 정보를 질의할 수 있는 방법이 필요하다. AMGA(Arda Metadata Grid Application) [1]는 그리드 상에 저장된 파일들에 대한 메타데이터에 대한 접근을 제공한다. AMGA는 PostgreSQL, MySQL, Oracle, SQLite와 같은 데이터베이스 시스템에 저장된 데이터를 접근하는 단순화된 인터페이스를 제공한다. 여러 응용들이 AMGA을 활용하고 있다. LHCb[2]는 북키핑(bookkeeping) 정보를 저장하기 위해 AMGA를 활용하고 있으며, Ganga는 작업들을 현황을 서술하는 메터데이터를 저장하기 위해 AMGA를 활용한다.

[1,4]는 AMGA 구현에 대한 성능을 평가하고 분석하였다. 이 논문들에서는 주로 논리 파일 카탈로그 서비스를 위한 LAN 및 WAN 환경에서 읽기와 쓰기 처리율을 측정하였지만, AMGA의 어떤 요소가 성능에 영향을 미치는지에 대한 설명이 없으며, 또한 직접 데이터베이스에 접근하는 것과의 성능차가 분석되지 않았다.

본고에서 우리는 AMGA 구현에서의 오버헤드가 무엇인지, 그리고 직접 데이터베이스 접근과의 성능차는 얼마나 되는지를 분석하기 위해 다시 성능을 측정하였다. 측정 결과를 보면 AMGA 구현에서 가장 큰 오버헤드는 GSI/SSL 연결을 처리하는 데에 있으며, 그 오버헤드가 약 350%정도 처리율을 저하한였다. 이 문제를 해결하기 위해 우리는 웹서비스 분야에서 널리 알려진 부하분산 기법을 AMGA 구현에 적용하였고, 그 결과 AMGA 서버의 수를 증가시켰을 때 그 수만큼의 처리율 향상이 가능하였다.

또한 우리는 그리드에서 실행 중인 작업들이 AMGA를 접근하는 경우를 가정하여 WAN 환경에서 AMGA 구현의 성능을 측정하였다. WAN 환경에서의 큰 네트워크 지연 시간 때문에 동시에 실행하는 AMGA 프로세스의 수가 전체 처리율에 가장 큰 영향을 주었다. 처리율을 증하시키기 위해 AMGA 프로세스들의 수를 증가시켰고, 이것은 뒷단의 데이터베이스 서비스가 AMGA 프로세스 수보다 많은 수의 프로세스를 갖게 하였고, 이것은 데이터베이스 시스템의 메모리 문제를 발생시켰다. 이 문제는 WAN 환경에서 AMGA 구현을 위한 부하

분산 기법을 적용하는 것을 어렵게 하였다. WAN 환경에서 이러한 시험 결과를 통해, 그리드 상의 작업이 직접 AMGA에 접근하는 경우 데이터베이스에 직접 접근하는 것만큼의 성능 향상을 얻기 위해서는 AMGA 내부에 DB 연결 풀링 기법과 부하분산 기법의 활용이 필요함을 확인하였다.

II. AMGA 성능 측정 및 분석

1. AMGA 오버헤드 측정

[1,4]는 논리 파일 카탈로그를 통해 AMGA 구현의 성능을 측정하였다. 본고에서도 동일한 서비스를 통해 AMGA 구현의 성능을 측정하였다. 이를 위해 우리는 논리 파일이름을 GUID(Global Unique ID)로 매핑하는 테이블을 생성하였다. 그리고 (파일이름. GUID) 쌍을 삽입하고, 파일이름으로 GUID를 질의하면서 성능을 측정하였다. 본고에서 모든 성능 측정은 상기 방법으로 진행하였다. 시험에서 우리는 각 삽입/질의의 전에 "Transaction" AMGA 명령을 실행하고, 삽입/질의 후에 "Commit" 명령을 실행하였는데, 이는 트랜잭션 처리 없이는 AMGA 명령들이 올바르게 실행되지 않았기 때문이다. AMGA에서 삽입/질의 명령은 실제로 데이터베이스에서 2개의 명령으로 변환되어 실행된다. 첫 번째는 AMGA가 데이터베이스에서 해당하는 테이블의 이름과 관련 정보를 얻기 위해서이고, 두 번째는 실제 삽입/질의 명령을 실행하기 위해서이다.

이 실험에서는 한대의 PostgreSQL(v 7.3) 서버, 한대의 AMGA(v 1.2.8) 서버, 그리고 8대의 AMGA 클라이언트 노드를 사용하였다. 이 시스템들은 모두 3GH Pentium D CPU와 2GB 메모리를 갖는 동일한 시스템들로써, 기가비트 네트워크를 통해 연결되어 있다. 우리는 PostgreSQL 설정파일에 max_connections=150, shared_buffers=8092, effective_cache_size = 163840로 설정하였고, AMGA의 주요파일에 MaxProcesses=130, Sessions=no, UseSSL=1로 설정하였다. 그리고 각 클라이언트 노드가 여러 개의 쓰레드를 동시에 실행하도록 하였고, 각 쓰레드는 계속하여 삽입/질의 연산을 통해 AMGA 서비스를 이용하여 로드(load)를 생성하도록 하였다.

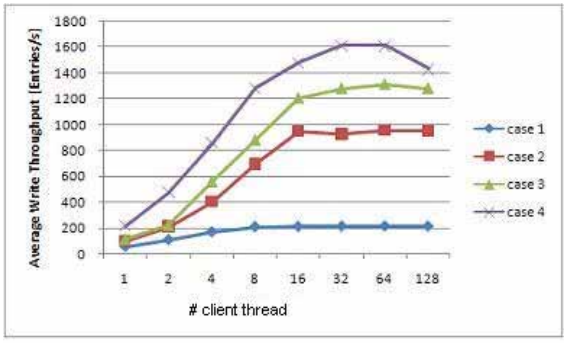
우리는 AMGA 구현에 포함된 오버헤드를 측정하기 위해 4개의 테스트 케이스를 준비하였다. 표 1의 Case1은 그리드에서 실행중인 작업들이 직접 AMGA 서비스를 접근하는 경우이다. Case1에서는 클라이언트 쓰레드가 AMGA 서버에 GSI/SSL 연결을 열고, 삽입/질의 연산을 한 다음, 연결을 닫는 과정을 반복한다. Case2는 GSI/SSL 연결을 사용하는 대신 일반 TCP/IP 연결을 사용한다는 점만 제외하면 Case1과 동일하다. Case1과 Case2 비교를 통해 GSI/SSL 연결을 처리하는 오버헤드를 측정하는 것이 가능하다. Case3은 몇몇의 주요 사용자나 프로그램들이 계속해서 AMGA 서비스를 활용하는 경우이다. Case2와 Case3의 비교를 통해 우리는 네트워크 연결을 열고 닫는 것을 처리하는 것에 대한 오버헤드를 측정할 수 있다. Case4는 AMGA에 접속하는 대신 데이터베이스에 직접 접근한다는 점만 제외하면 Case3과 동일하다. Case3와 Case4의 비교를 통해 우리는 상기 요소들 이외에 AMGA 서비스에 존재하는 오버헤드를 측정할 수 있다. Case4를 위해서 우리는 Case3을 실행했을 때 데이터베이스에 남겨진 로그로부터 어떠한 SQL 명령이 실행되었는지를 추출하였고, 같은 SQL 명령들을 직접 데이터베이스에 접속하여 실행하도록 하였다.

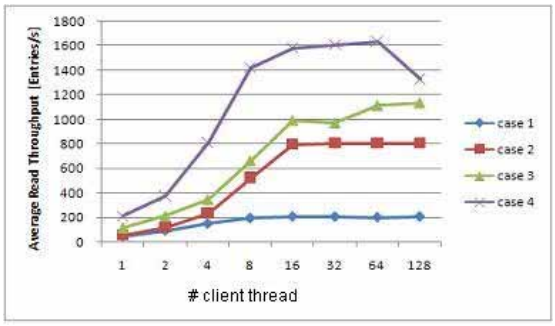
AMGA 구현의 삽입/질의 처리율을 알아내기 위해, 우리는 PostgreSQL의 모니터링 기능을 이용하였다. PostgreSQL 데이터베이스와 연결하여 모니터링 정보를 추출하는 pgtop 툴을 이용하면 초당 처리된 트랜잭션의 수를 알수 있다. 한 삽입/질의 연산은 한 트랜잭션을 유발하므로, 처리된 트랜잭션의 수를 모니터링함으로써 AMGA 서비스의 전체 처리율을 얻어낼 수 있었다.

[표 1] AMGA 오버헤드 측정을 위한 시험 방법

	시험 방법
Case1	AMGA를 통한 데이터베이스 접근 한 엔트리의 삽입/질의에 매번 신규 연결 GSI/SSL 연결 활용
Case2	AMGA를 통한 데이터베이스 접근 한 엔트리의 삽입/질의에 매번 신규 연결 GSI/SSL 연결 미활용
Case3	AMGA를 통한 데이터베이스 접근 여러 엔트리 삽입/질의에 한 연결 활용 GSI/SSL 연결 미활용
Case4	AMGA없이 데이터베이스 직접 접근 여러 엔트리 삽입/질의에 한 연결 활용 GSI/SSL 연결 미활용

그림 1은 AMGA 구현의 오버헤드 측정 시험의 결과를 보여준다. 우리는 한 클라이언트 노드에서 최대 16개의 쓰레드가 실행되도록 하였다.



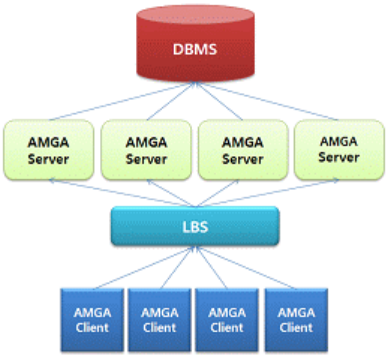


▶▶ 그림 1. 표1의 시험들에 대한 AMGA 처리율

그림 1에서 32 쓰레드란 2 노드들에서 동시에 32 쓰레드가 실행되었음을 의미한다. 가장 큰 오버헤드는 GSI/SSL 연결을 처리하는 데 있었으며, 직접 데이터베이스를 접근하는 것에 비해 약 350% 정도의 처리율 저하를 유발하였다. 두 번째로는 네트워크 연결을 열고 닫는 부분이 약 270% 정도의 처리율을 저하시켰으며, 기타 AMGA 서비스의 다른 기능들에서 약 80%의 처리율이 저하되어서, 전체적으로 직접 데이터베이스에 접근하는 것에 비해 약 700% 정도의 처리율 저하를 보였다. Case1을 보면 8개 쓰레드를 가진 한 클라이언트 노드가 AMGA 서버의 CPU를 100% 활용하도록 하였다. Case4에서는 128 쓰레드를 실행한 8개 클라이언트 노드가 뒷단의 DB 서버로부터 성능 저하를 유발하였다. 그리고 삽입 연산과 질의 연산은 처리율에 있어서 큰 차이를 보이지 않았다.

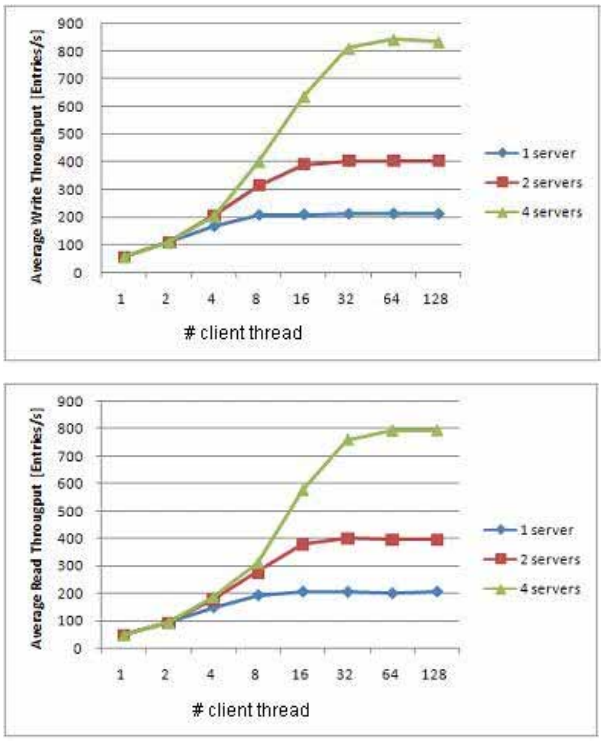
2. 부하분산 다중 AMGA 서버

그림1은 SSL/GSI 연결과, 네트워크 연결 처리가 AMGA 구현의 성능에 가장 큰 영향을 끼치는 요소라는 것을 보여주었다. 이 문제를 해결하기 위해 우리는 웹 서비스 분야에서 자주 활용되는 부하분산 기법을 시험하였다. 그림 2에서처럼 우리는 하나의 뒷단 PostgreSQL DB 서버, 4 AMGA 서버, AMGA 서버들 사이의 로드밸런싱을 담당하는 하나의 LVS[5] 서버, 그리고 4 클라이언트 노드들로 구성된 새로운 시험 환경을 구축하였다. 모든 하드웨어 사양과, 소프트웨어 설정은 1절과 같다.



▶▶ 그림 2. 부하분산 AMGA 서버스 시험환경

AMGA 서버의 수와 클라이언트 쓰레드의 수를 증가시키면서, 우리는 AMGA 서버에게 가장 큰 부하를 주었던 표1의 Case1의 경우에서 전체 AMGA 서비스의 처리율을 측정하였다. 1절에서처럼 각 클아이언트 노드마다 16개의 쓰레드를 실행하였고, 그림3에서 32 쓰레드는 2 클라이언트 노드에서 32개의 쓰레드가 실행되었음을 나타낸다. 그림 3의 128 쓰레드의 경우에만 각 노드당 32개의 쓰레드를 실행하도록 하였다.



▶▶ 그림 3. 부하분산 AMGA 서비스 처리율

그림 3은 삽입/질의 두 연산 모두에서 AMGA 서비스의 전체 처리율이 AMGA 서버의 숫자에 비례한다는 것을 보여준다. 우리가 보유한 리소스의 부족으로 4개를 초과하는 AMGA 서버에 대한 시험을 계속할 수 없었다. 그러나 우리는 부하분산 기법을 활용한 다중 AMGA 서버를 활용하는 방법을 통해 직접 데이터베이스에 접근하는 것만큼의 성능향상이 가능할 것으로 예측한다.

3. WAN 환경에서 AMGA 성능 분석

AMGA 서비스는 시작되면 미리 정해진 수만큼의 프로세스들을 실행한다. 그리고 더 많은 수의 AMGA 서비스 요청이 있는 경우, 미리 정해진 최대 프로세스의 수만큼 동적으로 AMGA 프로세스들이 추가로 실행된다. 한 AMGA 프로세스가 시작되면 먼저 뒷단의 데이터베이스에 연결을 맺고, 사용자에 대한 서비스를 시작한다. AMGA 프로세스와 뒷단의 데이터베이스와의 연결은 한번 시작되면 프로세스의 종료까지 끊

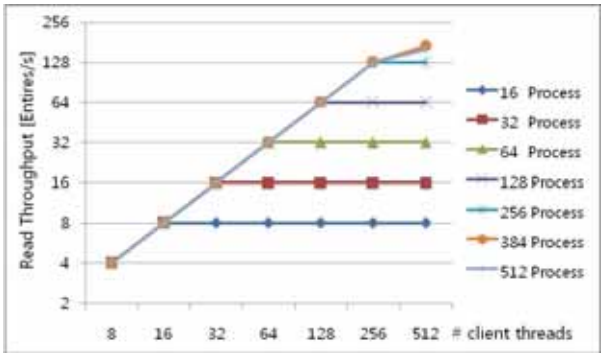
이지 않고 지속된다. 한 AMGA 프로세스는 한번에 한 사용자로부터의 연결만을 처리할 수 있어서, AMGA 프로세스의 수는 항상 동시 처리할 수 있는 동시 사용자 수를 나타낸다.

WAN 환경에서는 큰 네트워크 지연 시간 때문에 한 AMGA 프로세스는 한 사용자의 요청을 처리하는 데 LAN에 비해 많은 시간을 소요한다. WAN에서의 RTT(tound trip time)를 300 millisecond으로 가정하고(프랑스 IN2P3와 KISTI간 측정된 RTT), LAN에서의 RTT를 0.1 millisecond으로 가정하면(KISTI 내부 시스템간 측정된 RTT), 한 AMGA 프로세스는 정해진 시간동안 WAN에 비해 LAN에서는 3000배 많은 사용자 요청을 처리할 수 있을 것이다. Case1의 경우 연결을 맺고, 삽입/질의 연산을 처리하고, 연결을 끊는데, tcpdump[5] 프로그램을 통해 모니터링 한 결과 7쌍 이상의 메시지들이 오고 간다. 이 것은 RTT를 300 millisecond로 가정한 경우 한 AMGA 프로세스가 2초에 하나의 사용자 요청을 처리한다는 것을 나타낸다. 그림2에서처럼 한 AMGA 서버가 초당 200여개의 사용자 요청을 처리하기 위해서는 400개 이상의 AMGA 프로세스가 실행되어야 함을 나타낸다. 그러므로 동시에 실행되는 프로세스의 수는 WAN환경에서 AMGA 구현의 성능에 가장 큰 영향을 주는 요인이다.

이를 증명하기 위해 우리는 WAN 환경에서 표1의 Case1의 성능을 측정하려고 하였다. 그러나 원거리 사이트에서 활용할 만한 시스템이 없어서, WAN 환경을 에뮬레이션할 수 밖에 없었다. 에뮬레이션을 위해 우리는 RTT를 300 millisecond로 가정하고, Case1과 같은 상황에서 한 엔트리의 삽입/질의에 7쌍의 메시지가 오고 간다고 가정하였다. 이 경우 한 엔트리의 삽입/질의에 약 2초 이상의 네트워크 지연시간이 있을 것이고, 에뮬레이션을 간단하게 하기 위해 우리는 클라이언트 쓰레드가 AMGA 서버와 접속 후에 sleep(2) 명령을 통해 2초간 쉬도록 하였다.

이 테스트를 위해 우리는 한대의 뒷단 PostgreSQL 서버, 한대의 AMGA 서버, 1대의 클라이언트 노드를 사용하였다. 모든 하드웨어 사양은 1절과 동인하며, PostgreSQL이 max_connections=520의 설정과, AMGA가 MaxProcesses =512의 설정을 갖도록 하였다. 그리고 한 클라이언트 노드에서 512개의 쓰레드가 실행되도록 하였고, 이것은 클라이언드 노드에 큰 부하를 주지 않는 것을 확인하였다. 과거 1절, 2절의 시험에서 삽입 연산과 질의 연산이 처리율에 있어 큰 차이를 보이지 않았기 때문에, 이 시험에서는 삽입 연산 대신 질의 연산만을 수행하도록 하였다.

그림 4는 표 1의 Case1에 대한 WAN 환경에서 질의 연산에 대한 처리율을 보여준다. 그림 4를 보면 16개의 AMGA 프로세스가 실행되는 경우 초당 최대 8개 엔트리가 처리되었다. 이것은 한 AMGA 프로세스가 한번에 한 클라이언트 쓰레드로부터의 요청만 처리할 수 있음을 보여준다. 384개의 AMGA 프로세스가 실행되는 경우 초당 약 170개의 엔트리가 처리되었고, 512개의 AMGA 프로세스가 실행되는 경우 초당 약 160여개의 엔트리가 처리됨을 발견하였다. 이것은 AMGA 프로세스의 수가 많아짐에 따라 AMGA 서버에서 프로세스간 컨텍스트 스위칭이 빈발한 것에 따른 오버헤드 때문일 것으로 추측한다.



▶▶ 그림 4. WAN 환경에서 AMGA 프로세스 수에 따른 AMGA 서비스 처리율

우리는 또한 WAN 환경에서 AMGA 구현에 부하분산 기법을 적용하려 하였다. 이 테스트를 위해, 한대의 뒷단 PostgreSQL 서버, 2대의 AMGA 서버, 한대의 LVS 서버, 2대의 AMGA 클라이언트 노드로 시험 환경을 구축하였다. 모든 하드웨어 사양은 다른 시험에서와 동일하고, PostgreSQL이 max_connections=800 설정을 갖도록 하고, AMGA 서버가 MaxProcesses=380 설정을 갖도록 하였다.

AMGA 서버를 증가시키는 것은 AMGA 프로세스의 수를 증가시킬 수 있어 결과적으로 부하분산된 다중 AMGA 서버를 사용함으로써 좀 더 나은 처리율을 보일 수 있을 것으로 기대하였다. 그러나 각 클라이언트 노드에서 380개의 쓰레드가 동작하여 두 노드에서 모두 760개의 쓰레드가 실행되었을 때, 뒷단의 데이터베이스 서버에서 메모리 문제가 발생하였다. AMGA 프로세스 수의 증가는 뒷단의 데이터베이스 서비스에서 프로세스 수의 증가를 초래하였고, 이것은 데이터베이스 서버에서 많은 양의 메모리가 소요되도록 하였다. 이 시험을 통해 우리는 현재 AMGA 구현으로는 WAN에서 부하분산 기법을 활용한 다중 AMGA 서버 활용 방법이 의미가 크지 않다는 것을 확인하였다. 데이터베이스 연결을 여러 프로세스나 쓰레드들이 공유하여 사용하는 풀링 방법이 AMGA 구현 내부에 적용한다면, 데이터베이스 연결 수가 줄어들 수 있고, 이를 통해 부하분산 다중 AMGA 서비스를 구축하여 최적의 성능을 이끌어 내는 것이 가능할 것이다.

III. 결 론

본 고에서는 첫째로 현재 AMGA 구현의 오버헤드를 분석하였다. 직접 데이터베이스에 접근하는 것과 비교하여 GSI/SSL 연결의 처리가 약 350% 정도의 처리율 저하를 가져왔고, 전체적으로 약 700% 정도의 오버헤드가 존재하였다. 두 번째로 부하분산 다중 AMGA 서버와 DB 연결풀링 방법을 AMGA 구현에 채용하는 방법을 제안하였다. 우리는 본 고에서의 분석과 제안 방법이 LFC[7]. Fireman[8] 등 다른 유사한 그리드 서비스들에게도 적용되어 많을 도움이 될 것으로 기대한다.

▌참 고 문 헌▐

[1] N. Santos and B. Koblitz, Metadata services on the grid, in Proc. of Advanced Computing and Analysis Techniques (ACAT'05), Zeuthen, Berlin, May 2005
[2] LHCb - http://cern.ch/lhcb/
[3] Ganga - Gaud/Athena and Grid Alliance, http://cern.ch/ganga/
[4] B. Koblitz, N. Santos, and V. Pose, "The gLite AMGA METADATA CATALOGUE", "http://www.physics.ox.ac.uk/users/cioffi/UKmetadata/paper.ps
[5] LVS Linux Virtual Server, http://www.linuxvirtualserver.org/
[6] tcpdump, http://www.tcpdump.org/
[7] J. P. Baud, S. Lemaitre: The LHC File Catalogue (LFC),. HEPIX 2005, Karlsruhe, Germany
[8] EGEE Design Team, EGEE Middleware Architecure and Planning, August 2004, CERN Document EGEE-DJRA1.1-476451-v1.0