병렬 정보 검색 시스템의 고장 포용성 향상 기법

강재호[a], 안현주[b], 정성원[b], 류광렬[b], 권혁철[b], 정상화[b]

[a]동아대학교 지능형통합항만관리연구센터
부산광역시 사하구 하단2동 840 지능형통합항만관리연구센터
Tel: +82-51-200-5521, Fax: +82-51-200-5523, E-mail: jhkang@pusan.ac.kr

[b]부산대학교 컴퓨터공학부
부산광역시 금정구 장전2동 산30번지 부산대학교 공과대학 컴퓨터공학부
Tel: +82-51-510-{3582, 2875, 2453, 2218, 2434}, Fax: +82-51-{517-2431, 510-2218, 510-2218, 517-2431, 517-2431}
E-mail: {gus, swjung, krryu, hckwon, shchung}@pusan.ac.kr

요약

인터넷과 같은 대량의 정보에 대응할 수 있는 고성능 정보검색시스템을 구축하기 위해서 지금까지는 고가의 초대형 컴퓨터를 주로 활용하여 왔으나, 최근 가격대 성능비가 높은 PC 클러스터 시스템을 활용하는 방안이 경제적인 대안으로 떠오르고 있다. PC 클러스터 시스템에서는 전체 작업을 개별 노드 즉 PC에 가능한 균등하게 분배함으로써 성능을 극대화 하고자 하는데, 하나 또는 그 이상의 노드에 문제가 발생하는 경우 전체시스템의 성능이 매우 저하되거나 정상적인 서비스를 제공하기가 어려워진다. 이러한 상황에서 고장 포용성의 달성은 1년 365일 지속적으로 운영되어야 하는 많은 응용분야에서 반드시 해결해야 하는 문제이다. 본 논문에서는 PC 클러스터를 활용한 병렬정보검색시스템에서 고장 포용성을 극대화하기 위하여 각 노드의 색인어 역파일을 이웃 노드에 효율적으로 중복하여 저장하는 방안과 이를 활용한 효과적인 병렬정보검색 방법을 제안한다. 대규모 말뭉치를 활용한 실험결과 본 논문에서 제시하는 고장 포용성 향상을 위한 색인어 역파일 중복 저장방안이 충분한 효율성과 실용성이 있음을 확인하였다.

주제어

병렬정보검색; PC 클러스터; 고장포용

서론

최근 인터넷의 보급이 급격히 확대됨에 따라 정보검색 시스템이 처리해야 하는 정보의 양과 사용자의 검색요구는 폭발적으로 증대하고 있다. 이러한 수요에 대응하기 위하여 대부분의 정보검색 서비스 전문업체들은 고가의 중대형 서버 또는 슈퍼컴퓨터를 사용하여 서비스를 제공하고 있다. 대표적인 예로 AltaVista는 수십 기가바이트의 주기억 용량을 가진 초대형 시스템을 사용하여 하루에 수백만 건의 검색연산을 하고 있지만, 이러한 고가의 컴퓨터는 거액의 외화 부담을 요구할 뿐 아니라 대규모 사업자가 아닌 경우에는 거의 사용이 불가능하다. 반면에, 저가의 PC들을 고속 네트워크로 연결함으로써 고성능의 병렬 시스템을 실현하는 PC 클러스터 구조는 정보량이 폭증하는 검색 분야뿐 아니라 성능과 함께 빠른 응답시간이 요구되거나 실시간 처리가 필요한 다양한 응용분야에서 저비용으로 시스템을 구축할 수 있는 대안으로 주목 받고 있다[1][2].

PC 클러스터 기반의 병렬 정보검색 시스템을 현실적으로 구축하여 운영하기 위해서는 크게 두 가지 주요 사항을 고려하여야 한다. 첫 번째는 개별 노드에 효과적으로 작업을 분배함으로써 병렬처리의 효율을 극대화하여야 한다는 점이며, 두 번째는 시스템을 운영하는 과정에서 발생할 수 있는 각종 문제 상황에서도 서비스의 중단 필요성이 최소화될 수 있도록 고장포용성을 제공하여야 한다는 점이다.

정보검색에서 병렬처리의 효율을 극대화하기 위해서는 검색대상 자료를 각 PC의 하드디스크에 골고루 분산저장mfje I/O의 병목현상을 최소화하고, 각 PC에서의 검색계산 부하를 최대한 균등화할 수 있는 방안을 찾아야 한다. 정보검색의 병렬화를 위한 초기의 연구 중 대표적인 것으로는 Stanfill의 임의적 분산방법[3]이 있으며, 복수 개의 디스크를 가진 한 컴퓨터에서 색인어 역파일을 분할하여 저장하는 방법과 환경에 따른 성능을 시뮬레이션으로 분석한 연구로 [4]가 있다. 이 후 문서별 그리고 색인어별 역파일 분할을 함께 고려한 연구로 [5]가 있다.

* 정보통신부지원 대학기초연구지원사업(정보통신기초기술연구지원사업)으로 부산대학교 컴퓨터 및 정보통신연구소에서 수행하였음

PC 클러스터 기반의 병렬 정보검색 시스템을 제안하면서 색인어 역파일의 효과적인 분산저장 방식을 소개한 최근의 연구로 [6]이 있다. 이 연구에서는 색인어간의 문서내에서의 동시등장 확률을 연관관계로 하여 디클러스터링(declustering) 기법으로 색인어 역파일을 분산저장함으로써 무작위적 분산저장보다 성능향상을 이룰 수 있음을 보인 바 있다.

PC 클러스터 기반 병렬정보검색 시스템을 실용적으로 활용하기 위해서는 이러한 성능측면에서의 최적화 이외에도, 하드웨어 결함 또는 소프트웨어 유지보수와 같은 원인으로 인한 일부 노드의 중지가 요구되는 상황에서도 지속적으로 운영할 수 있는 고장포용성이 일정 수준 이상 보장되어야 한다. 최근들어 통신 인프라 및 하드웨어의 결함률이 낮아지는데 비해, 운영되고 있는 소프트웨어의 문제점 해결 또는 추가 서비스의 개발과 같은 시스템의 유지보수 요구가 더욱 빈번해지는 상황에서는 이러한 결함포용성의 중요성은 더욱 커진다고 할 수 있다.

데이터베이스분야에서는 초기부터 이러한 연구가 활발히 진행하여 왔으나[7][8], 병렬정보검색 분야에서는 아직 미진한 상황이다. 병렬정보검색 분야의 특성을 반영하여 이러한 문제에 접근한 연구로는 [9]가 있으며, 이 연구에서는 색인어별로 역파일을 이웃노드에 중복하여 분산저장하는 방안을 제안함으로써 성능의 향상과 고장 포용성 달성을 함께 해결하고자 하였다. 하지만 이 연구에서는 색인어별 역파일 분할을 염두에 두고 제안하였기 때문에, 시스템의 전반적인 유지보수와 확장성에 좀더 유리한 문서별 역파일 분할을 채택한 병렬정보검색 시스템에는 활용되기 어렵다.

본 논문에서는 문서별 역파일 분산저장에 기반한 병렬정보검색시스템에 실용적인 수준의 고장포용성을 제공하기 위하여 색인어 역파일을 이웃 노드에 중복하여 효과적으로 저장하는 방안과, 이를 활용한 동적 부하균형화 방안을 함께 제시한다.

본 논문의 구성은 먼저 2장에서 정보검색에 대한 일반적인 설명을, 3장에서는 병렬화를 위한 색인어 역파일의 분산저장방안에 대하여 설명한다. 이어지는 4장에서는 병렬정보검색의 고장 포용성 달성을 위하여 본 논문에서 제시하는 문서별 역파일의 분산중복저장기법과 이를 활용한 병렬정보검색 방안을 소개한다. 본 논문에서 제시하는 기법들을 활용한 실험결과를 5장에서 분석하고, 마지막 6장에서는 결론 및 향후연구과제를 제시한다.

정보검색

정보검색[10]은 사용자가 빠른 시간 내에 직접 파악하기가 어려운 방대한 양의 데이터에서 사용자가 요구하는 질문의 답으로 적절한 내용을 추출하고 가공하여 제시하는 컴퓨터 응용분야이다. 인터넷 정보검색과 신문기사 검색 등 현재 실용적으로 활용되고 있는 정보검색분야에서는 구축된 문서 및 이들 문서간의 연결 형태로 이루어진 데이터를 대상으로 사용자가 질의를 입력하면, 검색시스템은 보유한 문서와 주어진 질의와의 적합정도를 계산하여 점수화하고 정렬하여 사용자에게 제시하는 형태로 구현되어 활용되고 있다. 본 논문에서도 이러한 일반적인 문서 정보검색모델을 바탕으로 한다.

문서 모델링과 색인어 역파일

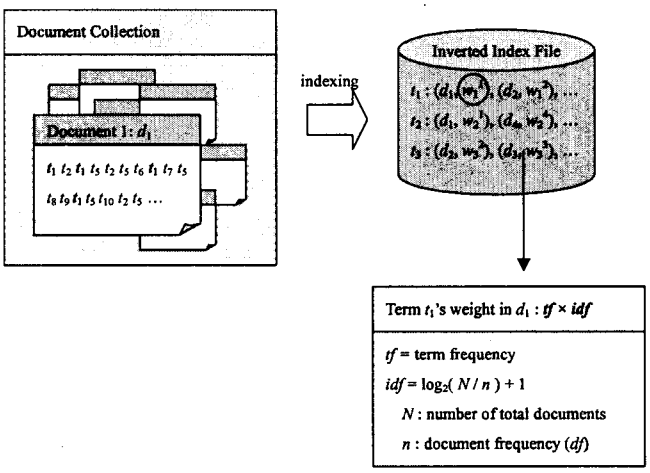

그림 1. 검색을 위한 색인어 역파일 생성

그림 1은 문서의 집합으로 이루어진 데이터를 검색에 효율적인 구조인 색인어 역파일(Inverted Index File) 형태로 가공하는 예를 보이고 있다. 하나의 문서는 등장 단어가 순서대로 나열된 데이터로 표현될 수 있다. 문서의 구조가 파악될 수 있는 경우에는 제목, 저자 또는 타 문서와의 연결관계와 같은 추가적인 정보도 포함하는 형태로 표현되어 보다 정확한 그리고 다양한 검색을 위한 자료로 활용되기도 한다 문서에 등장하는 단어들은 동사 및 형용사 등의 원형을 찾는 스테밍(stemming) 과정과 대부분의 문서에 등장하기 때문에 검색 시 효용가치가 없는 in, the와 같은 단어를 제거하는 불용어(stop words) 제거 과정을 거쳐, 색인에 사용할 수 있는 색인어의 나열로 표현한다. 한국어 문서의 경우 여기에 형태소분석을 위한 단계가 추가로 필요하다.

문서를 정보검색에서 보편적으로 활용되고 있는 순서를 고려하지 않은 색인어의 집합형태로 모델링하는 경우, 문서는 색인어 하나 하나를 축으로 하는 다차원상의 벡터로 표현될 수 있다. 이 때 특정 색인어 축에서의 문서벡터의 길이는 해당 색인어의 문서 내에서의 중요도와 전체 문서군에서의 중요도를 함께 고려할 수 있는 $tf \times idf$를 활용한 표현이 가장 널리 활용된다. $tf \times idf$는 문서 내에서의 해당 색인어

의 상대적인 중요도인 *tf* (term frequency) 항목과 전체 문서군에서의 해당 색인어의 중요도를 표현하는 *idf* (inter-document frequency)를 각기 계산하여 곱한 값이다. *tf*는 색인어가 해당 문서에서 얼마나 중요한지를 표현하는데, 문서에 많이 등장할수록 그 색인어가 해당 문서에서 중요하다는 가정하에 색인어의 등장 횟수로 계산한다. 보유한 문서들간의 길이 차이가 심한 경우에는 *tf*값을 개별 문서에서 최다등장색인어의 등장횟수로 정규화하여 사용할 수도 있다. *idf*는 전체 문서군에서 개별 색인어의 중요성을 추정하기 위하여 사용되는데, 많은 문서에 등장하는 색인어일수록 보다 보편적인 색인어로 간주되어 그 중요도가 낮도록 전체문서수를 해당 색인어가 등장하는 문서수로 나눈 값이다. 등장 문서수에 따라 *idf* 값이 급격하게 변하는 것을 방지하기 위하여 밑수가 2인 log를 취하는 경우가 일반적이며, 최소값을 보장하기 위하여 상수값을 더할 수도 있다.

색인어 역파일은 색인어별로 등장 문서 및 각각의 문서에서의 해당 색인어의 가중치를 파일에 기록한 형태로 질의로 주어지는 색인어가 나타나는 문서에 대한 정보를 효율적으로 찾을 수 있도록 구성한 것이다.

질의 모델링과 유사도 계산

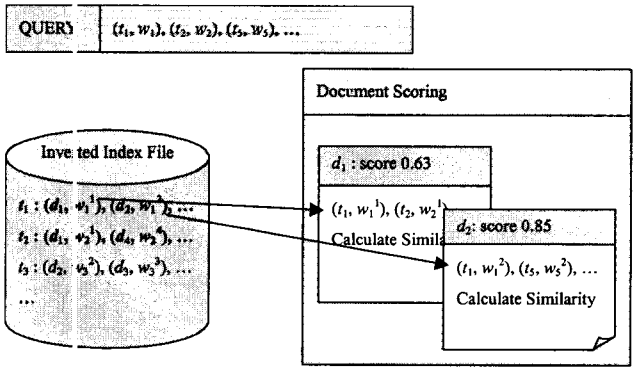

그림 2. 색인어 역파일을 이용한 검색

질의가 주어졌을 때 색인어 역파일을 이용하여 관련문서를 찾고 문서간의 우선순위를 매기는 방법에 대해서는 그림 2에 도시하고 있다. 먼저 질의를 문서와 동일한 형태로 표현한다. 즉 질의를 사용자가 제시한 질의 내 각 단어를 색인어로 하는 문서로 표현한다. 이렇게 함으로써 질의는 일종의 문서로 간주되며, 정보검색은 문서로 표현된 질의와 유사한 문서를 찾아내는 문제로 변환될 수 있다. 문서간의 유사도 비교방법으로는 벡터 공간상의 두 벡터의 각도를 계산하는 코사인 유사도(cosine similarity)를 이용하는 방법이 널리 사용되고 있다. 이러한 질의모델링 및 유사도 계산 방법은 기본적으로 질의와 유사한 형태로 색인어들이 분포된 문서를 선호하게 된다.

질의어의 가중치로 음수값을 취할 수 있게 하여 해당 질의어가 가능한 등장하지 않는 문서를 선호하도록 표현할 수도 있다. 보다 복잡하고 정교한 질의를 지원하기 위하여 AND, OR, NOT 등의 논리연산자(Boolean operator)나 NEAR등의 인접연산자(proximity operator)를 허용하는 검색시스템도 상당수 있다. 이러한 질의에서 논리연산자는 문서의 선택여부를 결정하는데 사용하고, NEAR 등의 인접연산자는 유사정도를 계산할 때 관여하게 된다.

적합성 피드백

사용자들이 질의를 만들어 필요로 하는 정보를 검색하는 경우, 습관적으로 자신이 찾고자 하는 정보와 관련된 소수의 단어만으로 질의를 표현하는 경우가 태반이다. 이에 비해 검색엔진에는 다양한 분야에 대하여 대규모의 문서집합을 보유하고 있음으로 해서, 해당 질의와 조금이라도 연관이 있는 문서는 상당한 분량이 된다. 이러한 상황에서 검색결과문서 중 질의와 높은 유사도를 가졌다고 시스템이 평가한 문서라고 해서 사용자가 요구하는 정보에 근접한다고 말하기가 어렵다. 즉, 적은 개수의 질의어로 질의가 표현됨으로써, 상대적으로 사용자가 원하는 정보와 그렇지 못한 정보를 구분하는 변별력이 떨어지게 되는 것이다. 이는 곧 사용자에게는 검색이 정확하지 않은 것처럼 보여지게 되며, 다양한 방법을 통하여 이를 보완하는 연구는 꾸준히 진행되어 왔다[11].

적합성 피드백(relevance feedback)[11]은 이러한 정확도를 개선하기 위한 방법중의 하나로, 검색된 문서 중에서 사용자가 문서의 제목, 요약 또는 전문을 직접 살펴본 후 적절 또는 부적절하다고 판단되는 문서를 선정하면, 시스템이 이들 문서를 반영하여 재검색함으로써 사용자의 의도에 보다 충실한 검색결과를 얻고자 하는 방법이다. 초기 검색결과 중 높은 점수를 받은 문서들은 상대적으로 사용자 요구에 어느 정도 부합하는 결과일 것이라는 가정하에 시스템이 자동적으로 이러한 재검색과정을 수행할 수도 있다. 이러한 적합성 피드백은 초기검색의 결과 문서들에 포함된 색인어들을 사용자 또는 시스템이 평가한 정도에 따라 가중치를 주어 추가함으로써 보다 복잡한 질의로 확장하고 새로이 생성된 질의로 검색을 다시 수행함으로써 구현된다. 적합성 피드백은 기존 검색시스템에 수정을 거의 요하지 않으면서 상당한 정확도를 추가로 얻을 수 있어 정보검색분야에서 널리 활용될 수 있는 방법이다.

본 논문에서 대상으로 하는 병렬정보검색시스템은 이상에서 설명한 문서와 질의를 벡터 공간상에 표현하고, 검색의 정확도를 향상시키기 위해 적합성 피드백 기능을 기본적으로 제공하는 시스템이다.

정보검색시스템의 작업부하

앞에서 살펴본 정보검색시스템에 질의가 주어졌을 때 발생하는 작업부하는 크게 색인어 역파일에 접근하여 필요한 정보를 읽어내는데 소요되는 디스크 I/O관련 작업과 질의와 문서간의 유사정도를 계산하는 작업이 주된 요소를 이룬다. 이들은 기본적으로 주어진 질의에 의해 검색결과로 발견한 문서의 수에 비례하여 작업량이 늘어난다. 즉 많은 문서에 나타나는 질의어가 질의에 포함되는 경우 디스크 I/O 및 유사도 계산을 위한 작업량이 그만큼 늘어나게 되며, 정보검색시스템이 보유한 총문서의 수가 많을수록 각 색인어가 등장하는 문서수 역시 이에 비례하여 늘어난다.

대략적인 정보검색시스템의 작업부하를 주요 항목으로 모델링하면 작업량은 $|D| \times |q| \times r$ 에 비례한다고 말할 수 있다. $|D|$는 정보의 양 즉 보유한 문서수이며, $|q|$는 질의에 나타나는 평균적인 질의어의 수이며, r은 사용자의 질의 요청 횟수이다. 서론에서 밝힌 바와 같이 검색 분야 특히 인터넷 검색의 경우에는 검색의 대상이 되는 문서 수가 급증하고 있으며, 사용자의 질의 요청도 더욱 빈번해지고 있다. 또한 보다 정확한 검색결과의 도출을 위한 방법들은 기본적으로 검색시스템의 부하를 상당히 가중시킨다. 본 연구에서 채택하고 있는 적합성 피드백과 같은 질의 확장기법의 경우 처음 사용자가 제시한 질의보다 수 배 내지 수십 배 많은 질의어를 가진 확장된 질의를 재처리해야 하므로, 검색시스템의 부하는 매우 높아지게 된다. 비록 컴퓨터 하드웨어의 가격은 지속적으로 하락하고 이에 비해 성능은 계속 높아지지만, 대단위의 문서를 대상으로 하는 검색 분야에서는 하나의 소형 시스템으로 감당할 수 있는 그 한계가 있음은 분명하다. PC 클러스터를 활용한 병렬 정보검색시스템은 이러한 지속적인 문서의 추가 유입과 사용자의 검색요구의 증가, 그리고 보다 정확도가 높은 검색결과를 제공하기 위하여 늘어날 수밖에 없는 작업량을 기존 클러스터에 추가의 PC를 편입시켜 확장하는 방법으로, 질의응답시간을 적절한 수준으로 계속 유지할 수 있는 현실적인 해결방법의 하나임은 틀림이 없다.

계속해서 3장에서는 이러한 PC 클러스터 상의 병렬정보검색시스템을 최대한 효율적으로 활용하기위한 역파일 분산저장기법에 대하여 설명한다.

병렬검색을 위한 색인어 역파일 분산저장

병렬 정보검색을 위하여 역파일을 분산 저장하는 방안은 그림 3에서와 같이 두 가지 방향에서 접근할 수 있다 문서와 색인어간의 관계를 2차원 행렬로 나타내었을 때, 첫번째는 색인어별로 즉 행 단위로 노드에 분산하여 저장하는 색인어별 역파일 분할 방법이며, 두 번째는 열 단위로 하나의 문서와 관련된 데이터를 하나의 노드에 저장하는 문서별 역파일 분할 방법이다. 그림에서는 4개의 노드를 가진 시스템으로 가정하였는데 색인어별 역파일 분할의 경우

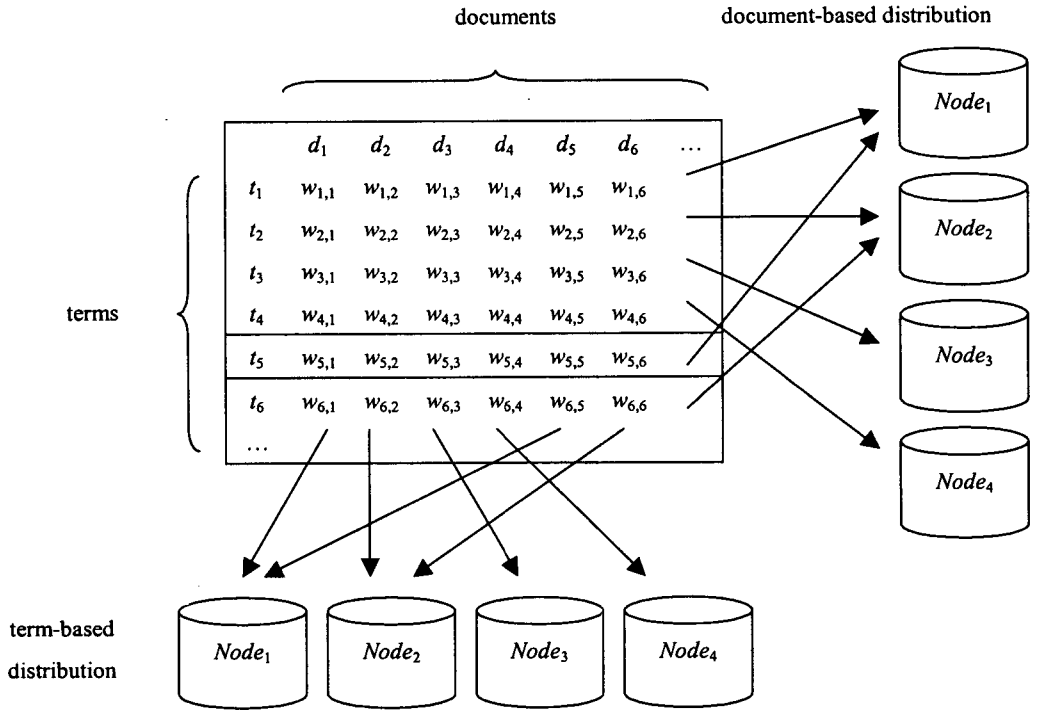

그림 3. 병렬정보검색을 위한 역파일 분할 방법

$Node_1$에는 색인어 (t_1, t_5, t_9, ...)을 $Node_2$에는 (t_2, t_6, t_{10}, ...)을 할당하는 방법으로 연관된 역파일을 분산 저장할 수 있다. 마찬가지로 문서별 역파일 분할의 경우 $Node_1$에는 문서 (d_1, d_5, d_9, ...)을 $Node_2$에는 (d_2, d_6, d_{10}, ...)을 할당할 수 있다.

색인어별로 역파일을 분할한 병렬정보검색시스템은 질의 처리 시 본래 역파일을 사용하는 경우와 동일한 횟수의 디스크 입출력 횟수를 보장 받으므로, 디스크 접근의 기본비용은(접근 횟수에 비례한다고 가정) 최소화된다. 하지만 적은 수의 단어로 이루어진 질의를 처리하는 경우 질의에 포함된 색인어를 보유하지 않은 노드는 작업을 할당 받을 수 없으므로 응답시간의 최적화에는 제한을 받게 된다. 예를 들어 t_2와 t_3 두 단어로 이루어진 질의를 처리하는 경우 $Node_1$과 $Node_2$는 관련 데이터가 없으므로 검색에 참여하지 않게 되고 이러한 유휴 노드는 전체적인 시스템의 성능 저하로 이어질 수 있다. 적합성 피드백과 같은 고급 질의 확장을 통하여 보다 높은 품질의 검색결과를 도출하는 시스템의 경우 기존 연구[9]와 같은 방안으로 이러한 유휴 노드를 가능한 줄일 수 있지만, 이 경우에도 수십 대 이상의 노드를 활용하는 초대형 정보검색시스템으로 발전시키기에는 그 한계가 있다 하겠다.

색인어별로 역파일을 분할하는 방안을 적용하였을 때 시스템의 확장이 어려운 보다 근본적인 이유는 개별 노드이서 처리된 검색결과를 모두 취합하여 문서별 부분점수를 모두 누적하고 전체정렬과정을 거쳐야 최종적인 검색결과를 도출할 수 있다는 점 때문이다. 색인어별 역파일 분산저장의 이러한 특성은 보유한 문서수에 비례하여 통신량이 증가하는 결과를 낳기 때문에, 인터넷 상의 수십억 단위의 문서를 대상으로 하는 정보검색시스템을 구축하기에는 상대적으로 과도한 통신소요량이 큰 부담이 될 수 있다.

문서별 역파일 분할 방법의 경우 각각의 노드는 자신에게 할당된 문서들에 대하여 색인어 역파일을 구성하고 질의가 주어지면 보유한 데이터에 대해서만 검색을 수행한다. 일반적으로 질의와 부합되는 문서 중 가장 높은 점수를 받은 N개의 문서에 대해서만 사용자가 관심을 가지므로 개별 노드에서 가장 점수가 높은 N개의 문서에 대한 정보만을 통신을 통해 취합하고 정렬한다. 문서별 역파일 분할 방식에서 이러한 노드별 독립성은 전체 통신비용을 보유 문서에 관계없이 일정 수준 이하로 제한할 수 있으므로 색인어별 역파일 분할 방식에 비하여 시스템의 확장성이 보다 우수하다고 할 수 있다.

고장포용을 위한 색인어 역파일 중복 저장

병렬 정보 검색 시스템을 실용적으로 활용하기 위해서는 효율성 증대를 위한 노력과 더불어 서비스 제공 중에 발생할 수 있는 노드의 결함이나 유지관리를 위해 일부 노드를 정지하여야 하는 극한 상황에도 적극적으로 대비할 필요가 있다. 특히 빠른 응답시간과 고성능을 달성하기 위하여 클러스터에 포함된 모든 노드가 전체 작업에 참여하는 PC 클러스터기반 병렬처리 시스템의 경우, 노드 하나의 결함은 전체 시스템의 운영중지 상황으로까지 이어질 수 있으므로 결함포용성에 대한 중요성은 더욱 커진다.

일부 노드의 사용이 불가능한 경우에도 시스템의 운영할 수 있는 가장 적극적인 대처방법은 전체 시스템을 둘 이상으로 복제하여 운영하는 방법이나, 이는 그 효율에 비하여 추가 비용이 과다하게 발생하므로, 본 논문에서는 작업분배의 직접 대상이 되는 색인어 역파일을 중복저장함으로써 시스템의 효율성과 결함포용성을 동시에 달성하는 방안을 제안한다. 병렬 정보검색 시스템에서 개별 색인어 관련 역파일 엔트리 정보를 2개 이상의 서로 다른 노드에 중복하여 저장한다면 특정 노드 결함 시에도 복사본을 이용하여 운영이 가능하며, 본 논문에서는 저장 공간상의 요구가 가장 낮은 2 노드 색인어 역파일 엔트리 중복저장방안을 연구하였다.

색인어 역파일 중복저장 방안

하나의 문서 관련 역파일 데이터를 2개의 서로 다른 노드에 중복저장할 때, 원본이 저장될 노드를 주노드, 사본이 저장될 노드를 부노드라고 부르기로 한다. 하나의 노드에서는 보유한 데이터 중 자신이 주노드인 문서를 주문서라고 하고 자신이 부노드인 문서는 부문서라고 한다. 색인어 역파일 정보를 중복저장하기 위한 부노드로는 그림 4에서와 같이 오른쪽 방향에 있는 노드를 선정한다. 즉, $Node_1$을 주노드로 선정한 문서와 관련된 색인어 역파일 정보는 $Node_2$에 중복 저장되며, $Node_2$에 주문서로 할당된 문서는 $Node_3$에 부문서로 중복 저장된다. 이와 같이 모든 문서와 관련된 정보를 2개의 노드에 중복하여 저장함으로써, 하나의 임의 노드 고장시에도 복사본을 이용하여 검색시스템을 계속 운영할 수 있게 된다. 또한, 이웃하지 않은 복수개의 노드에 문제가 발생하는 경우에도 운영을 지속할 수 있다.

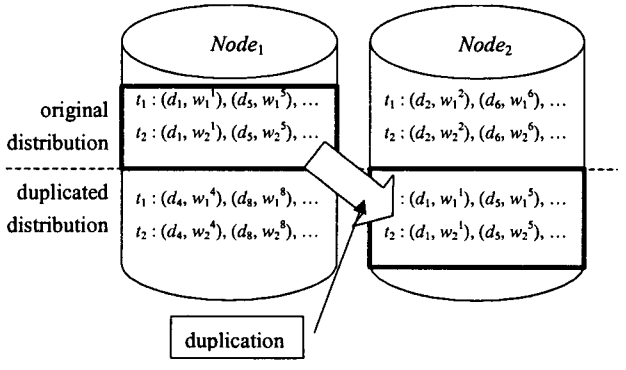

그림 4. 문서별 색인어 역파일 중복 저장의 예

그림 5에는 총 8개의 노드 중 $Node_2$와 $Node_6$에 문제가 발생한 경우를 보이고 있다. 각각 $Node_3$과 $Node_7$에 중복된 데이터를 이용하여 운영이 가능함을 보이고 있다. 이러한 복수 개의 노드에 동시에 문제가 발생하는 상황은 우연적으로 일어나기는 어려우나, 소프트웨어 및 하드웨어의 유지 보수를 위하여 일부 노드를 정지시켜야 하는 상황에는 효과적으로 대처할 수 있다는 점에서 그 의의가 있다 하겠다.

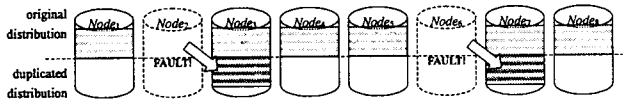

그림 5. 문제 발생 시 운영 방안

색인어 역파일 중복 저장을 활용한 병렬 정보검색

색인어 역파일을 중복저장하고 문제가 발생한 노드가 보유했던 데이터를 이의 복제본을 가진 노드가 처리하도록 하는 단순한 해결방안 즉, 정적부하균형화 방안은 그림 5과 같은 경우 $Node_3$과 $Node_7$에 평소보다 2배의 부하가 걸리게 한다. 이러한 노드간의 부하불균형은 사용자에 대한 서비스 반응속도 측면에서 바람직하지 않다. 따라서 문제가 발생한 경우 추가로 발생하는 부하를 가능한 여타 노드에 균등하게 분배함으로써 보다 효율적으로 시스템을 운영할 수 있는 방안을 강구할 필요가 있다. 그림 6에는 $Node_2$와 $Node_6$에 의해 야기되는 추가 부하를 각각 $Node_3$, $Node_4$, $Node_5$그리고 $Node_7$, $Node_8$, $Node_1$번이 균등하게 ⅓씩 나누어 분산하여 처리함으로써 노드간의 부하편차를 최소화할 수 있는 동적 부하균형화 방안의 예를 보이고 있다.

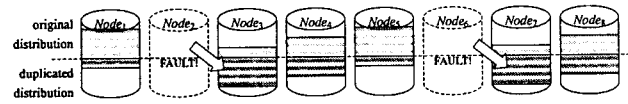

그림 6. 문제 발생 시 효과적인 운영 방안

이러한 방식으로 노드에 문제 발생시 추가되는 부하를 효과적으로 분산하기 위해서는 특히 디스크 I/O 횟수를 최소화할 수 있도록 색인어 역파일 구조를 변경할 필요가 있다. 그림 7은 이를 위하여 본 논문에서 제안하는 색인어 역파일 중복 저장 방안을 설명하고 있다. 각각의 색인어에 대한 색인어 역파일 엔트리는 자신의 주문서와 부문서에 대한 데이터를 디스크상에 연이어서 저장한다.

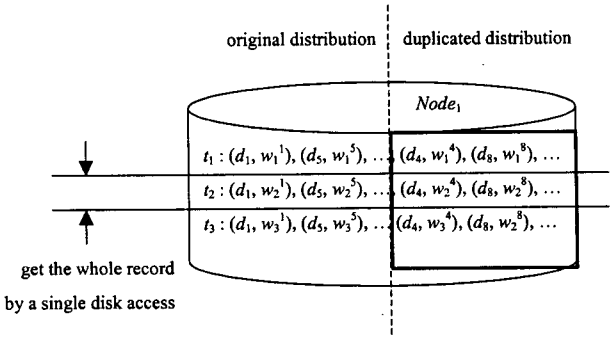

그림 7. 역파일 중복 저장 구조

그림 8에는 하나의 노드에서 주문서 및 부문서 관련 데이터에 접근하기 위한 색인어 역파일을 저장하고 인덱스를 유지하는 예를 보이고 있다. 색인어 t_i에 대한 색인어 역파일 레코드는 주문서에 대한 정보와 부문서에 대한 정보를 디스크 상에 연이어서 가지고 있으며, 이를 위하여 인덱스 구조에도 각각의 디스크 내 위치를 기입하여 유지한다. 정상적인 상황에서는 주문서에 관한 데이터만을 이용하여 중복저장을 하지 않은 경우와 동일한 방법으로 검색을 수행한다.

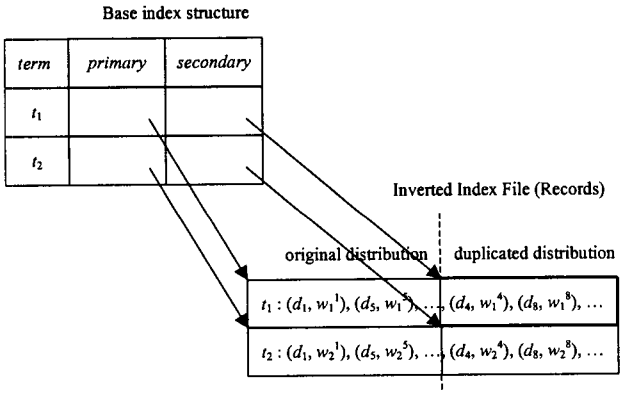

그림 8. 중복저장시 역파일 인덱스 구조

그림 9는 전체 4개의 노도로 이루어진 PC 클러스터 기반 병렬정보검색 시스템에서 $Node_1$에 문제가 발생하였을 때 나머지 $Node_2$, $Node_3$, $Node_4$가 처리하여야 하는 문서의 범위를 색인어 역파일 레코드에서 굵은 사각형 영역으로 보이고 있다. 노드에서 처리하여야 하는 범위는 문서번호 영역을 노드별로 지정하는 방법으로 할당이 가능하다. 하지만, 색인어별로 해당 레코드에서 특정 번호의 문서의 위치를 정확하게 파악하는 것은 과도한 인덱스 구축 및 유지 비용을 요구하기 때문에 현실성이 없으며, 본 연구에서는 이 위치를 어림잡는 방안을 적용한다.

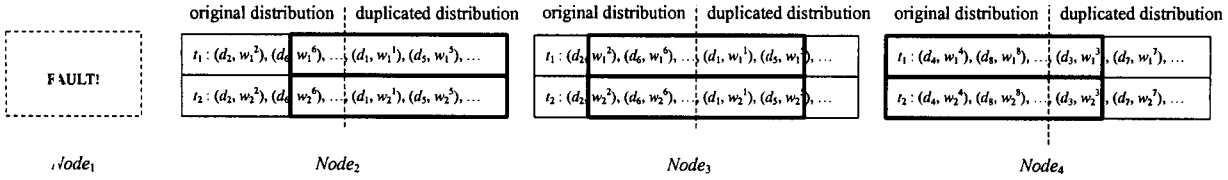

그림 9. 문제 발생시 역파일 접근 및 처리 방법

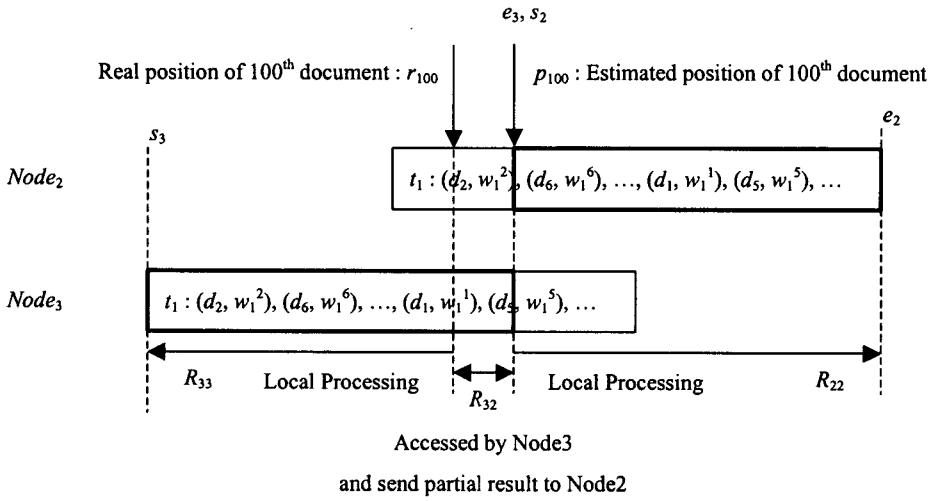

그림 10. 문제 발생시 역파일 접근 및 처리 방법

Constructic n Step:
For each tern t_i in term set T
  Allocate a term t_i to a node n_j with a predefined method
    (e.g. round robin)
ParallelFor i n each node n_j
  Construct inverted index file iif_j for n_j

Retrieval S.ep:
Get a user's query q
Broadcast the query q to all nodes
ParallelFor i n each node n_j
  Process the query q at node n_j with local inverted index file iif_j
  Make partial scored documents list P_j which relevant to query q
Gather all partial scored documents P_j list from all nodes
  (The comm.. cost is proportional to the # of total documents)
Make global score list G for each documents in P_j
  by accumulating partial scores
Find global best-N documents from global score list G by sorting
Present the global best-N documents list to the user

그림 11. 중복 저장 및 병렬검색 알고리즘

그림 10에 이러한 예를 보이고 있는데, 여기서는 $Node_1$에 문제가 발생하였으며 이를 $Node_2$와 $Node_3$에 균등하게 분배하여야 하는 경우를 가정하고 있다. 전체 문서의 수가 200개라 가정한다면 $Node_2$는 $Node_1$의 주문서 즉 자신의 부문서를 모두 처리하고, 자신이 주노드인 문서 중 101번~200번 문서에 해당되는 데이터 즉 절반에 대해서 처리한다. $Node_3$은 $Node_2$에서 중복저장된 부문서 데이터 중 중 1번~100번 문서에 해당되는 문서와 자신이 주노드인 문서를 모두 처리함으로써 $Node_2$와 $Node_3$은 균등하게 부하를 분배할 수 있다. 이 때 대략적으로 100번 문서에 대한 위치로 $Node_2$와 $Node_3$이 공통으로 가지는 색인어 역파일 레코드에서 중간지점을 p_{100}로 선정하였다 색인어 t_1의 분포가 예상과는 달리 후반에 수집된 문서에 보다 자주 등장하는 경우 그림에의 r_{100}과 같이 100번 문서는 역파일 레코드의 앞부분에 치우쳐 위치하게 된다. $Node_2$는 이러한 내용에 대해 알 수 없으나, $Node_3$은 접근한 색인어 역파일 부분에서 문서 번호를 확인함으로써 자신이 담당이 아닌 영역 R_{32}를 발견하게 되고, 이를 $Node_2$에 전송하여 처리하도록 한다. 그림 11에는 이러한 문서별 색인어 중복저장 방안을 이용하였을 때의 병렬정보검색 알고리즘을 기술하고 있다.

2차 검색을 위한 중복 저장

2차 검색은 질의와 문서간의 코사인 유사도로 검색(1차 검색) 결과로 추출된 상위 점수의 문서를 대상으로 질의에 적합한 문서내 문장 탐색, 문서 요약, 검색 결과 문서의 클러스터링과 같은 고급 연산을 수행함으로써 사용자가 보다 정보를 쉽게 이해하고 탐색할 수 있도록 지원하는 기능이다. 대부분의 정보검색 시스템은 질의에 적합한 문장 제시 수준 정도는 이미 제공하고 있으며, 보다 고급화된 검색결과를 사용자에게 제공할 수 있는 방안에 대해서도 많은 연구가 이루어지고 있다. 2차 검색은 대부분 원문 또는 원문에 가까운 데이터를 활용하여 수행되며, 1차 검색에 대한 부가적인 서비스가 아니라 기본 서비스의 하나로 활용되는 추세이므로 이 또한 정상적인 서비스의 안정적인 운영을 위해서는 고장 포용성을 반드시 필요로 한다고 할 수 있다.

각각의 문서는 1차 검색을 위한 구조에서 선정된 주노드와 부노드에 중복하여 저장되며, 색인어 역파일 구조와는 달리 원문에서의 단어의 위치 정보도 함께 포함되어 개별 문서 단위로 접근가능한 구조로 관리된다. 1차 검색과 마찬가지로 일부 노드에 문제가 발생하는 경우 뿐 아니라 정상상황에서도 2차검색 대상문서의 수가 적어 특정 노드에 부하가 집중될 수 있으므로 효과적으로 부하를 분산할 수 있는 방법이 필요하다. 2차 검색은 문서 단위로 디스크 접근이 일어나므로 상대적으로 수월하게 동적 부하균형화를 수행할 수 있으며, 본 연구에서는 가능한 붐빌 것으로 예상되는 노드에 할당될 가능성이 높은 문서를 먼저 처리하기 위하여 각 노드들의 부하를 예측하고 이를 참고하여 문서의 우선순서를 선정한 후 greedy하게 문서를 할당하는 방법을 개발하였다. 이에 대한 알고리즘은 그림 12에 표기하였다.

Construction Step:
For each document d_i in documents set D
  Allocate a document d_i to a node n_j with a predefined method
ParallelFor in each node n_j
  Construct inverted index file iif_j for n_j

Retrieval Step:
Get a user's query q
Broadcast the query q to all nodes
ParallelFor in each node n_j
  Process query q at node n_j with local inverted index file iif_j
  Make local scored documents list L_j which relevant to query q
  Make local best-N documents from L_j by sorting
Gather local best-N documents list L_j from all nodes
Find global best-N documents
  from all local best-N documents list L_j G by sorting
Present the global best-N documents list to the user

그림 12. 2차 검색을 위한 동적부하균등화 알고리즘

실험 결과

이상에서 설명한 색인어 역파일 분산 및 중복저장 방안의 효과를 검증하기 위해 일련의 실험을 수행한 결과를 이 장에서 정리하였다. 실험을 위한 병렬 컴퓨팅 환경으로는 8대의 PC를 Gigabit Ethernet 기반의 고속 네트워크로 연결한 PC 클러스터 시스템을 활용하였다. 각각의 PC는 800MHz 펜티엄 III CPU, 512MB 주기억장치 그리고 SCSI 방식의 하드디스크를 장착하였다. 소프트웨어 환경으로는 Linux를 탑재하였으며 메시지 기반 통신방식인 MPI를 구현한 MPICH를 사용자 레벨 통신 환경인 VIA 상에 동작하도록 수정한 MVICH[12]를 사용하였다.

실험 대상 말뭉치로는 10년 간의 신문기사 약 200만 건의 모음을 사용하였다. 실험에는 각각 24개의 색인어를 가진 500개의 질의가 사용되었고 이들을 이용하여 병렬 검색을 수행하고 질의 하나를 처리하는데 요구된 평균 시간을 5회 반복 실험 후 평균을 취하여 비교하였다. 질의 각각은 적합성 피드백을 가정하여 임의로 96개 이상의 색인어를 가진 문서를 선정하고 그 문서에서 *tf×idf* 값이 높은 순으로 24개의 색인어를 선택하는 방법을 사용하였다. 질의는 모두 OR 연산을 가정하였다. 2차검색 방법으로는 각 문서의 문장 단위로 질의와의 연관성을 코사인 유사도와 질의어에 해당되는 색인어간 거리를 동시에 반영한 모델을 개발하여 사용하였다.

먼저 본 논문에서 제시한 중복저장과 부하균형화 방안의 효과를 살펴보기 위하여 정상상황 및 문제가 발생한 상황에서 전반적인 성능을 평가하였다. 실험에 사용한 방법은 1차 및 2차검색에서 적용한 부하균등화 방안에 따라 SS, SD, DD의 세가지로 나누었다. 방법 SS는 1차 및 2차검색 모두 정적부하균등화 방안을 이용한 경우이며, SD는 1차 검색은 정적, 2차 검색은 동적 부하균등화 방안을 적용한 경우이다. DD는 1, 2차 검색 모두 동적부하균등화 방안을 적용한 가장 유연한 경우이다. 세가지 방법 모두 검색과 관련된 데이터를 중복저장하고 있다.

그림 13에는 이들 각각의 방안에 대하여 정상상황, 한 노드 고장, 두 노드 동시 고장 상황에서 질의 하나를 처리하는데 평균적으로 소요되는 시간을 보여주고 있다. SS 방법의 경우 노드에 문제가 발생한 경우 해당 노드에서 처리하여야 하는 작업을 이웃노드(부노드)에서 처리하여야 하므로 수행에 소요되는 시간이 상당히 증가하였다. 이에 비해 SD와 DD 방법은 추가 부하를 나머지 노드들이 적절히 분산하므로 성능의 저하가 작은 것을 알 수 있으며, 특히 DD 방법은 1차 검색에서도 이러한 분산 작업을 효과적으로 수행하므로 가장 성능이 우수함을 볼 수 있다. 정상상황에서 SS와 SD/DD 간의 수행시간의 차이는 2차검색의 동적부하균형화에 의해서 처리시간이 단축된 결과이며 문제가 발생한 상황에서 SD

와 DD간의 성능 차이는 1차검색에서 동적부하균형화의 효과이다.

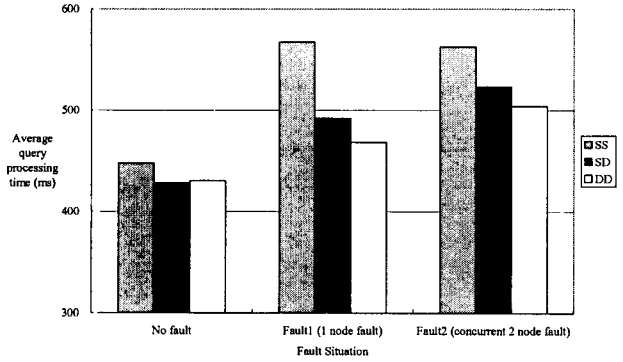

그림 13. 정상 및 문제 발생 시 시스템의 성능

1차 검색에서의 중복저장을 이용한 동적 부하균형화 방안이 얼마나 효과적인지 살펴보기 위하여 각각의 노드에서 처리한 평균 문서수를 살펴보았다. 그림 14는 2개의 노드($Node_2$와 $Node_6$)에 문제가 발생하였을 때 각 노드별로 1차 검색에서 처리한 문서의 수이다. SS/SD 방법은 $Node_2$, $Node_6$의 부하를 그대로 $Node_3$과 $Node_7$이 이어 받아 처리하여야 하므로 다른 노드에 비해 2배 분량의 작업량이 할당된 것을 알 수 있다. 이에 비해 DD 방법에서는 전체적으로 고르게 부하가 평준화 되었음을 알 수 있다. 그림에서 DD-Local과 DD-Neighbors는 DD 방법에서 읽어 들인 색인어 역파일 데이터 중에서 각각 자신이 담당한 부분과 이웃 노드에 전송된 데이터의 분량이다 이러한 추가적인 통신을 필요로 하는 데이터는 전체 분량에서 큰 비중을 차지하지 않았으며 (6% 이하), 같은 데이터 분량인 경우 고속 통신을 통하는 것이 디스크에 다시 접근하여 읽는 경우보다 비용이 적게 들기 때문에 그림 13에서 보듯 DD 방법이 DS 방법에 비해 우수한 성능을 나타낸 것으로 해석된다.

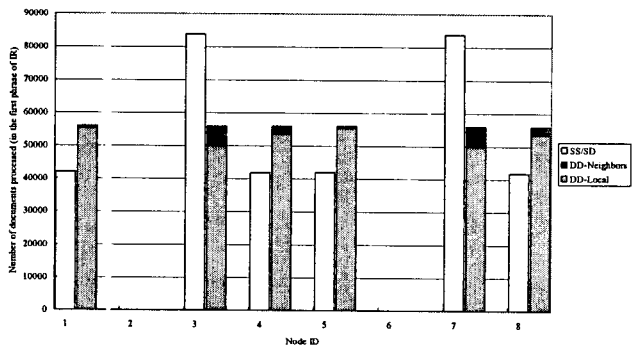

그림 14. 2 두 노드에서 동시 문제 발생시 노드별 평균 문서 처리 수 (1차검색)

그림 15는 2차 검색시 중복저장 구조를 이용한 동적 부하균등화 방안의 효과를 보여주고 있다. 1차 검색결과 중 상위 점수를 받은 128개의 문서가 2차검색의 대상으로 설정하였다. SS 방법은 앞에서와 같이 2차 검색시 문제가 발생한 노드 2번과 6번 노드의 부하를 각각 노드 3번과 노드 7번이 모두 담당하게 되어 매우 불균형한 부하 분배 상황을 보여주고 있으며, 이에 비해 2차 검색시 동적부하균형화를 수행하는 SD/DD 방법은 고른 부하균형상황을 보여준다.

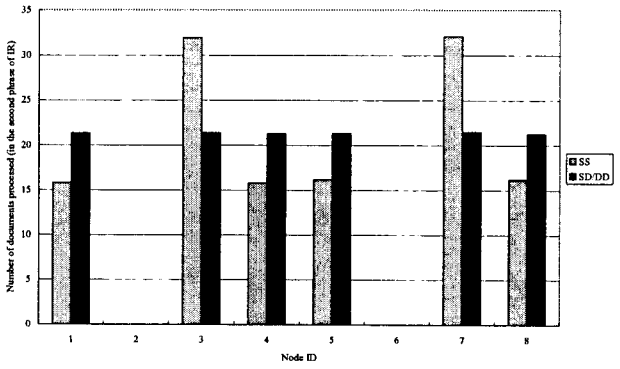

그림 15. 2 두 노드에서 동시 문제 발생시 노드별 평균 문서 처리 수 (2차검색)

이상과 같은 실험을 통하여 본 연구에서 제안한 중복저장 구조를 도입함으로써 병렬정보검색 시스템의 고장 포용성을 향상시킬 수 있음과 이러한 저장 구조의 특성을 반영한 동적부하균형화 방안이 고장 발생 시에는 시스템의 성능 저하를 최소화할 수 있음을 보였다.

결론 및 향후과제

본 논문에서는 PC 클러스터 기반의 병렬 정보검색 시스템의 고장포용성을 안정적으로 제공하기 위하여 색인어 역파일을 PC 클러스터의 노드에 중복하여 저장하는 기법을 연구하였다. 1차검색을 위해서는 색인어 역파일을 두 개의 노드에 중복 저장하되 문제 발생시 디스크 접근 횟수를 최소화할 수 있도록 디스크상에서 하나의 색인어에 대한 주문서 및 부문서에 대한 역파일 레코드를 인접하여 위치시켰다. 이러한 구조를 효율적으로 활용할 수 있는 동적 부하균형화 방안을 함께 제시하였다. 점차 중요시되는 2차검색과 관련해서는 고장포용성과 더불어 부하균형화를 함께 달성할 수 있는 저장방식과 문서할당 방안을 제안하였다. 10년간 200만건의 대규모 말뭉치를 대상으로 한 실험결과 본 연구에서 제안한 저장방식 및 활용방안이 충분한 실용성이 있음을 확인하였다. 향후 PC 클러스터를 이용한 병렬정보검색 시스템을 지속적으로 확장할 때 고성능의 신규노드를 도입하거나, 업그레이드 후 발생할 수 있는 노드간

의 성능차이에 의한 부하불균형을 실시간에 탐지하고 이에 효과적으로 대처할 수 있는 적응성 있는 부하균형화 방안을 추가적으로 연구할 필요가 있다.

참고문헌

[1] Lin, Z., and Zhou, S. (1993) Parallelizing I/O intensive applications for a workstation cluster: a case study, *Computer Architecture News* 21, 5, pp. 15-22

[2] Samanta, R., Zheng, J., Funkhouser, T., Li, K. and Singh, J. P. (1999) Load Balancing for Multi-Projector Rendering Systems, *SIGGRAPH / Eurographics Workshop on Graphics Hardware*, August

[3] Stanfill, C. and Thau, R. (1991) Information Retrieval on the Connection Machine : 1 to 8192 Gigabytes, *Information Processing & Management*, pp. 285-310

[4] Jeong, B. and Omiecinski, E. (1995) Inverted File Partitioning Schemes in Multiple Disk Systems, *IEEE Transactions on Parallel and Distributed Systems*, 6(2) pp.142-153

[5] Sornil, O. and Fox, E. A. (2001) Hybrid partitioned inverted indices for large-scale digital libraries, *Proceedings of The 4th International Conference of Asian Digital Library*, Bangalore, India, Dec. 10-12

[6] Chung, S-H., Kwon, H-C., Ryu, K. R., Jang, H-K., Kim, J-H. and Choi, C-A., "*Parallel Information Retrieval on an SCI-Based PC-NOW*," Lecture Notes in Computer Science Vol. 1800, (IPDPS-2000 Workshops, Cancun, Mexico) pp. 81-90, 2000.

[7] Wolfson, O., Jajodia, S., and Huang, Y. (1997) An Adaptive Data Replication Algorithm, *ACM Transactions on Database Systems*, Vol. 22, no. 2, pp. 255-314

[8] Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996) The dangers of replication and a solution. *Proceedings of ACM SIGMOD '96*. pp. 173-182

[9] 강재호, 양재완, 정성원, 류광렬, 권혁철, 정상화 (2003), 효율적인 병렬정보검색을 위한 색인어 군집화 및 분산저장기법, *정보과학회논문지* Vol. 23 No. 2

[10] Baeza-Yates, R. and Ribeiro-Neto, B.. *Modern Information Retrieval*. Addison-Wesley, Wokingham, UK, 1999

[11] Salton, G. and Buckely, C. (1990) Improving retrieval performance by relevance feedback, *Journal of the American Society for Information Science* 41, pp. 288-297

[12] National Energy Research Scientic Computing Center (1999) MVICH - MPI for Virtual Interface Architecture
http://www.nersc.gov/research/FTG/mvich/