Quick Sort 를 이용한 Ada Tasking 특성 분석
The analysis of Ada Tasking Behavior by Quick Sort

김민성 (멀티미디어과)
Min-Sung Kim (Dept. of Multimedia)

Key Words : Task, Package, Quick Sort, Rendezvous, Object Oriented Design, Para Information Hiding, Entry, Accept, Communication Sequential Process.

ABSTRACT : Ada reflects the concept of software engineering really well, and a functions, using task, this paper analyzes the features of parallel processing, real reliance in Ada through the Quick Sort algorithm. It also embodies data abstraction b correlates with the method of object oriented design.

1. 서론

소프트웨어의 위기 인식이 대두된 배경에는 하드웨어의 급격한 발전에도 불구하고 생산 및 유지 보수해야하는 소프트웨어 시스템의 규모가 증대하고 복잡해지는 속도에 비해 이를 구현하는데 필요한 방법론 확립과 도구들의 개발 등, 소프트웨어 개발 환경이 미비한데 기인되었다. 이와 같은 문제를 해결 하기 위한 방안으로써 수많은 소프트웨어의 개발 방법론과 자동화 지원 도구 및 프로그래밍 언어가 FORTRAN 과 COBOL 이후 꾸준히 개발되었지 궁극적인 생산성, 이식성, 신뢰성 등의 소프트웨어 제반 문제의 극복에는 빈약할 따름 이였다.

미국 국방성(department of defense) 은 이러한 소프트웨어의 위기를 인식하고 소프트 개발 환경의 스폰서로 나서 1969년 조립식 컴퓨터의 구성과 강력한 컴퓨터 언어 및 지원 시스템의 제정을 추진하여 Ada 프로그래밍 언어와 Ada 프로그래밍 지원환경을 개발하였다.

Ada는 소프트웨어 시스템의 이식성, 명확성, 신뢰성, 효율성 및 관리 유지 가능성을 제고하여 소프트웨어의 개발 방법론, 자동화 지원도구, 설계 및 프로그래밍 언어로써 수명주기(life-cycle) 전체를 지원하고 다 차원적인 특징을 가지고 있어 일반적인 응용분야 컴퓨터 시스템의 실시간 병행처리 및 인공지능 분야에도 사용될 수 있다.[1,4]

Ada가 가지고있는 특성 중에 가장 강력한 기능 중의 하나로써 태스크(task)를 이용 병렬처리는 real time 시스템을 구현 할 수 있다는 장점을 가지고 있다. 본 논문에서는 정 방법 중 가장 효율성이 뛰어난 알고리즘 중의 하나인 quick sort를 이용하여 ada tasking 의한 병렬처리 및 실시간 처리를 구현해 보고자한다.

2. Ada Task 구조

실세계(real world)의 문제 공간에는 많은 행위들이 동시에 논리적으로 일어난다. 이 문제 소프트웨어적으로 해결하기 위해서 기존의 고 수준 언어로써는 해결 방법을 찾기가 어렵다. 단지 host 컴퓨터의 운영체제나 멀티 태스킹 어셈불리 루틴의 사용이 가능하게 해 줄 뿐이었다. 그러나 이 대안은 프로그램의 이식성을 극도로 저하해주고 어셈블리 언어의 사용은 태스크간의 상호작용의 제어를 신뢰성있게 해주지 못했고 타이밍 관계의 표현을 분명하게

해주지 못했다. Ada는 프로그래밍 언어 자체가 병렬처리 기능을 가지고 있기 때문에 프로그램의 이식성과 프로그램 작성의 용이함을 가능하게 해 준다.

Ada에서 태스크는 다른 프로그램 단위와 병행으로 운영되는 단위를 말한다. 태스크는 논리적으로는 여러 개의 독립된 태스크로 구성되는 소프트웨어를 뜻하고 물리적으로는 멀티 컴퓨터 시스템, 멀티 프로세서 시스템 또는 단독 프로세서 내에서 수행되고 실세계에서 병행활동의 표현에 아주 적합하게 사용될 수 있다.

3. Ada Task 응용

Task는 명세와(specification) 와 몸체(body)로 구성이 된다. 피호출(called) task의 경 부분은 entry를 통해 다른 task와 접속을 위해 사용되고 호출(calling) task의 경우, 부분은 사용하지 않는다. 본 논문에서는 task간의 상호작용을 printer, adder 및 sorter 태스크를 사용하여 다음과 같이 응용해 보았다.

```
with text_io;
with quicksort;

procedure AdaQuickSort  is
   use text_io;
   package int_io is new integer_io(integer);
   use int_io;
   use quicksort;

   A       : vector;
   lower : integer := 1;
   upper : integer := n;


procedure start is

      task printer is
         entry s_print;
         entry a_print(sum:in integer);
      end printer;

      task adder is
         entry add;
      end adder;

      task sorter is
         entry sort;
      end sorter;
```

```
    task body printer is
      r_sum:integer;
    begin

      sorter.sort;
      accept s_print;

      accept a_print(sum:in integer) do
        r_sum := sum;
      end a_print;

    end printer;

    task body adder is
      sum:integer;
    begin
      accept add;

      sum:=0;
      for i in 1..upper loop
        sum:= sum + A(i);
      end loop;

      printer.a_print(sum);
    end adder;

    task body sorter is
    begin
      accept sort;
      sorting(A, lower, upper);
      printer.s_print;
      adder.add;
    end sorter;


  begin
    null;
  end start;
begin
  start;
end AdaQuickSort
```

Task들은 서로 통신하고 피호출 task의 명세 부분은 entry를 통하여 그 task와 통신허가 원하는 다른 task와의 접속을 제공한다. 피호출 task의 명세부분의 entry는 다른 task와 통 설정되는 지점으로써 피호출 task의 몸체 내부의 critical point인 accept 문이 시작 지점이다

프로그램이 시작과 동시에 위의 세 태스크인 printer, adder 와 sorter는 동시에 실행이 printer의 s_print와 a_print, adder의 add 그리고 sorter의 sort는 각각의 entry 부분으 대응하는 피호출 task의 신호를 기다린다. Entry 선언과 같이 사용되는 accept 문은 피호 task의 몸체에 위치하여 피호출 task가 다른 호출 task와 만나는 지점을 나타낸다. Acc 문은 accept로 시작하는 것을 제외하고는 그것에 일치되는 entry 선언과 동일한 형식 갖는다. 피호출 task의 명세 부분의 entry 선언과 몸체 부분에서의 accept 문은 협력하 동작한다. Entry 선언은 호출 task와 피호출 task를 동기화 시켜주고 accept 문은 실 task를 외부 세계에 개방하여 task가 다른 task와 동시에 수행 되게 하고, 정보교환 가능하게 하는데 이렇게 함으로써 랑데부(rendezvous)가 이루어지게 된다.[3]

위 프로그램에서 피호출 task printer와 호출 task sorter는 각각 랑데부 지점(rende point)에 도달할 때까지 독립적으로 수행하며, printer는 sorter가 랑데부 지점에 도달 때까지 대기 상태(waiting)에 있게된다. 호출 task가 도달할 때 두 태스크는 동기 화된 랑데부가 끝나면 두 task는 분리되어 독립적으로 계속 수행한다.

Task printer 본체의 sorter.sort는 accept sort 문에서 대기하고 있다. Task 호출하여 sorter는 package에 의해 선언된 sorting을 실행시킨다. Sorting에 의해 정렬된 A는 task sorter의 task printer 호출에 의해 프린트되고 제어 권은 다시 sorter로 sorter는 task adder를 호출하고 프로그램은 종료된다.

Task sorter의 본체에서 호출된 procedure sorting은 독립된 package 안에서 선언이 이 함수는 정렬 방법중의 하나인 퀵 정렬(quick sort)로써 Ada 특성 중의 하나인 task를 다 활용하고 있다. 먼저 Ada의 package 와 task 특성을 활용한 퀵 정렬의 구조를 분석해 본다.

4. 퀵 정렬의 구조

정열 방법 중에서 평균 적인 수행 시간이 가장 효율적인 퀵 정렬은 기본 배열의 한 값을 주축 값(pivot)으로 설정하여 배열을 다시 하는 것이다. 정렬의 기본적인 개념은 주축 값을 중심으로 그보다 적은 값의 모든 원소는 주축 값의 아래쪽에 위치하도록 설정하고 주축값 보다 같거나 큰 값을 가지는 원소는 위쪽으로 위치시킨다. 퀵 정렬은 기본적으로 다음 세 단계를 기초로 한다.[2,5]

1) Divide: 파일 A[p..r]을 두 부분 리스트(sublist) A[p..q]와 A[q+1..r]로 분활 (part 리스트 A[p..q]에서 각각의 원소는 리스트 A[q+1..r]의 모든 원소보다 적거나 원소 A[q]는 분활 하기 위한 주축값으로 사용한다.
2) Conquer:두 분할된 리스트 A[p..q]와 A[q+1..r]은 다시 퀵 정렬 알고리즘을 사용하여 정렬한다.
3) Combine:모든 분활된 리스트들이 순환 호출(recursive call)에 의하여 정렬이 되기 때문에 합병이 필요하지 않는다. 전체 파일 A[p..r]이 차례로 정렬이 되어있는 상태다.

퀵 정렬 함수는 다음과 같다.

```
QUICKSORT(A,p,r)
1 if p<r
2    then q ← PARTITION(A,p,r)
3         QUICKSORT(A,p,q)
4         QUICKSORT(A,q+1,r)
```

위 알고리즘의 핵심은 함수 partition에 있다.

```
PARTITION(A,p,r)
1 x ← A[p]
2 I    p - 1
3 j    r + 1
4 while TRUE
5    do repeat j ← j - 1
6          until A[j] ≤ x
7       repeat i ← i + 1
8          until A[i] ≥ x
9       if I < j
10         then exchange A[i] ↔ A[j]
11         else return j
```

각각의 단계를 순차적으로 표현하면 그림 1과 같다. 여리게 색칠된 원소는 정확한 분활(partition)이 된 후의 모습이고 진하게 색칠된 원소는 아직 분활(partition) 되기 전의 모습

(a) 초기의 array 모습이다. i와 j는 처음과 끝 원소에서 한 레코드씩 array의 밖을 가리킨 분활(partition)은 x=A〔p〕=5를 주축값으로 시행한다.
(b) i와j의 처음 while loop 시행후의 위치이다.
(c) partition 함수의 10번째 줄에서의 새로운 i와 j의 위치이다.
(d) i와 j의 두 번째 while loop 시행후의 위치이다.
(e) i와 j의 세 번째 겸 마지막 while loop 시행후의 위치이다. 함수가 종료되고 i 〉=j, 값이 return 된다.

5. Ada에 의한 퀵 정렬의 구현

Ada 시스템은 하나이상의 프로그램 단위(program unit)에 의해서 구성이 되고 각 구성 개별적인 컴파일이 가능하다. 프로그램 단위는 부 프로그램(sub-program), 태스크(ta 패키지(package) 및 범위단위 등이 있으며 모든 단위들은 선언부(specificatio 실현부(body)로 이루어져 각 단위 안에서의 분리 컴파일도 가능하다. 태스크는 앞에서 살펴본

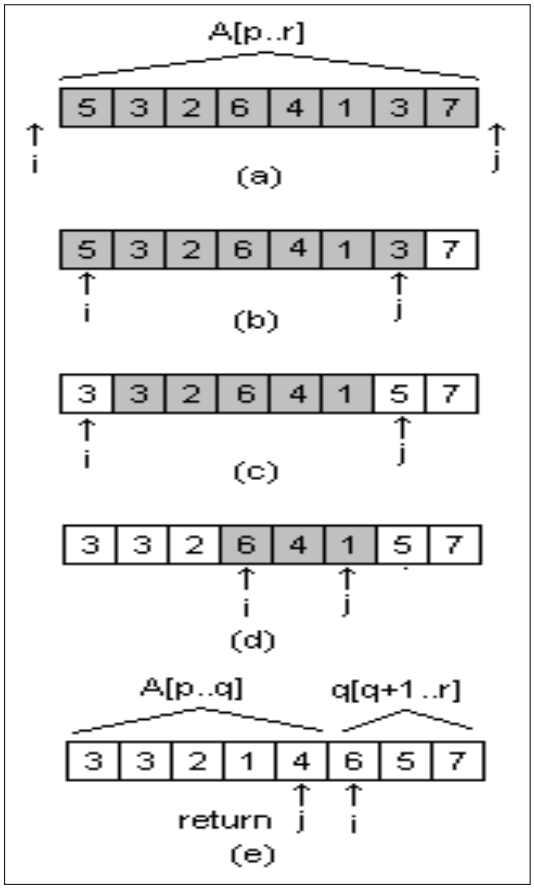

그림 1. 퀵 정렬 과정

바와 같이 논리적으로 병행 수행되는 행위의 정의에 사용되고 패키지는 계산 자원의 집합체로 필요에 의해 외부에서 사용할 수 있어 프로그램시 모듈과, 추상화, 축소와(localization) 및 정은폐(information hiding) 기능을 제공한다. 다음은 패키지 안에 선언된 퀵 정렬이다.[4]

```
package quicksort is

   n : constant := 30;
   subtype num_range is integer range 0..1000;
   type vector is array (1..n) of num_range

   procedure exchange(x,y: in out integer);
   procedure split(A:in out vector; lower, upper:
                   in integer;  half: in out integer);
   procedure two_part(A:in out vector; lower,
                   upper, half:in integer);
   procedure sorting(A:in out vector; lower, upper:
                   in integer);
```

```
end quicksort;


package body quicksort is

   ------------------------------------------
   -- Array의 두 원소를 교환한다.
   ------------------------------------------
   procedure exchange(x,y: in out integer) is
      temp: integer;
   begin
      temp:=x;
      x:=y;
      y:=temp;
   end exchange;

   ------------------------------------------------
   -- Array를 두 부분으로 분활한다.
   ------------------------------------------------
   procedure split(A:in out vector; lower, upper:
                        in integer; half:in out integer) is
      m,
      i,j,
      first,middle,last:integer;

   begin
      if upper - lower = 1 then
         if A(lower) < A(upper) then
            m:= A(lower);
         else m:= A(upper);
         end if;
      else
         first := A(lower);
         last := A(upper);
         middle := A( (lower+upper)/2);

         if first < middle then
            if first < last then
               if middle < last then
                  m:= middle;
               else m:= last;
               end if;
```

```
        else m:= first;
        end if;
      else if middle < last then
             if first < last then
               m:=first;
             else m:=last;
             end if;
           else m:=middle;
           end if;
      end if;
   end if;

   i:= 1;
   j:= upper;

   while i<j loop
      while A(i) <= m loop
        i:=i+1;
      end loop;

      while A(j) > m loop
        j:=j-1;
      end loop;

      if i<j then
        exchange(A(i), A(j));
      end if;
   end loop;

   half:= j;

end split;

-------------------------------------------------------------
-- 핵심이 되는 procedure이다. 분활된 두
-- array를 각각 task를 사용하여 퀵 정렬을
-- 순환호출 (recursive call) 하였다.
-------------------------------------------------------------
procedure two_part(A:in out vector; lower, upper,
                   half:in integer) is

   task lower_part;
```

```
    task body lower_part is
    begin
      sorting(A, lower, half);
    end lower_part;

    task upper_part;

    Task body upper_part is
    begin
      sorting(A, half+1, upper);
    end upper_part;

  begin
    null;
  end two_part;

  ------------------------------------------------------------
  -- 퀵 정렬이 불려지는 초기 procedure이다.
  ------------------------------------------------------------
  procedure sorting(A:in out vector; lower,upper:
                    in integer) is
    half: integer;

  begin
    if lower = upper then
      return;
    else split(A, lower, upper, half);
         two_part(A,lower,upper,half);
    end if;
  end sorting;

begin
  null;
end quicksort;
```

퀵 정렬을 패키지화 하여 네 개의 procedure, exchange, split, two_part 그리고 s 선언하였다. sorting은 직접적으로 외부 태스크인 sorter에 의해 호출이 된다. 앞 AdaQuickSort 프로그램에서 세 태스크인 printer, adder 그리고 sorter는 entry의 사용 호출 태스크의 의사표현을 비호출 태스크가 수락하여 두 태스크는 다시 독립적으로 각자의 일을 수행한다. 이는 명시적인 동기화 방법인 랑데부에 의하여 한 태스크가 상대편 태스크와

랑데부를 원하는 지점에 도달했을 때 다른 태스크가 이를 감지하여 이루어진다. 반면 패키지 quicksort에 의해 선언이된 two_part procedure안의 두 태스크, lower_part와 upper_part는 CSP(communication sequential process)로 다루었다. 이 방법에 의해서 태스크간의 통신에 있어서 동기화를 필요에 의해서 자연스럽게 이룰 수 있다.

6. 결론

Ada의 기능 중의 하나로써 태스크는 실시간 처리와 병렬 처리를 요구하는 소프트웨어 시스템 개발에 탁월한 능력을 갖고 있다. 태스크란 병렬처리 될 수 있는 프로그램 단위를 말하며 이것을 사용하면 실세계에서 동시에 일어나는 사건들을 모델링 하는데 매우 유용하다. 비트에 의한 실수의 계산을 하지 않고 실수 자체의 계산을 함으로써 더욱 쉽게 결과를 얻는 것과 같이 현실의 순차적 방법에 의한 일 처리 대신 태스크를 사용함으로써 실생활에서 일어나는 동시적 상황들을 실제와 극히 유사한 형태로 추상화 할 수있다.

본 논문에서는 태스크의 동기화 방법인 랑데부(rendezvous)를 퀵 정렬의 응용에 의해 구현해 보았고 패키지의 사용으로 데이터 추상화 기능을 표현하여 소프트웨어 시스템 설계시, 다루는 대상을 중심으로 하여 모듈을 분리하는 대상 중심형 설계(object oriented des 방법을 이용해 보았다.

랑데부에 의하여 태스크간 커뮤니케이션을 마치 인간이 대화하듯이 서로 시간적, 공간적으로 동기화 시켜주었다. 이 모형에서 만일 또 다른 태스크가 랑데부 지점에 이르기 전에 태스크가 진입 또는 수락 할 준비가 되어있는 상태라면, 특정 시간 주기를 기다리게 되거나, 커뮤니케이션 하기 위한 준비 상태의 또 다른 태스크를 진입시키거나 수락 할 수도 있다. 태스크를 사용함으로써 갖는 큰 장점 중의 하나가 커뮤니케이션의 신뢰성을 높여 주는데 있다. 태스크가 활동을 못하여 지연되면 또 다른 태스크가 비활동 태스크를 검출하고 적합한 대책을 마련 할 수가 있다. 태스크 간의 동기 화는 태스크 커뮤니케이션 관계를 아주 쉽게 표현해준다.

참고문헌

(1) 김 용진, 1988, “미래의 언어 Ada", 전자과학, 제 30권, 제 347호, pp. 130~153.

(2) 한 상영, 1990, 자료구조론, 와이 제이 출판부, 서울, pp. 334~339.

(3) 이 병복, 1993, “Ada Tasking 실행시간 복잡도의 효율적인 분석을 위한 ATSN에 관한 연구”, 전북대학교 석사논문, pp. 10~13.

(4) Barnes J. G. P., 1989, *Programming in Ada*, Addison-Wesley Publishing Company, Inc., pp. 277~301.

(5) Gormen, T. H.외, 1991, *Introduction to Algorithms*, The MIT Press, pp. 153~171.