2000 년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 1 호

병행성 이론에 기반한 자바 라이브러리 설계

신현삼*, 김재우**, 권기항***
*동아대학교 컴퓨터공학과
e-mail : sapip@nownuri.net
e-mail : kwon@se.donga.ac.kr

Design of Java library base on pi-calculus

Hyun-sam Shin*, Jae-woo Kim**, Kee-hang Kwon*
*Dept. of Computer Engineering, Donga-A University

요 약

자바에서 또한 이러한 병행성 프로그램을 지원하기 위한 다양한 언어적 지원과 병행적인 요건을 지원하고 있다. Thread class 지원, Synchronized 예약어, 상호협동 관계를 표현하기위한 wait(), notify(), notifyAll() 메서드 , monitor 메커니즘등을 지원하고 있다. 하지만 이는 아주 저 수준의 지원형태이며 여러 가지 문제점을 야기할 수 있다. 이에 대해 병행적 이론의 정수라고 할 수 있는 pi-calculus 의 이론과 기법을 도입하여 새로운 시각으로 병행성 프로그래밍에 대한 접근을 하고자 한다. 여기에 실용적으로 많이 사용되고 발전되어온 자바에서 적용하고자 한다. pi-calculus 에 기반한 pict 를 모델로 하여 pict 에서 지원하는 아주 명료한 연산자와 current object 를 도입함으로써 보다 표현력과 확장성, 검증성이 뛰어난 고 수준 자바 라이브러리를 설계하고자 한다.

1. 서론

최근 병행성 프로그래밍의 연구가 하드웨어의 비약적 발전과 소프트웨어의 대형화 추세에 활발히 연구되고 있다.

자바에서 또한 이러한 병행성 프로그램을 지원하기 위한 다양한 언어적 지원과 병행적인 요건을 지원하고 있다. Thread class 지원, Synchronized 예약어, 상호협동 관계를 표현하기위한 wait(), notify(), notifyAll() 메서드 , monitor 메커니즘등을 지원하고 있다. 하지만 이는 아주 저 수준의 지원형태이며 여러 가지 문제점을 야기할 수 있다. 이에 대해 병행적 이론의 정수라고 할 수 있는 pi-calculus 의 이론과 기법을 도입하여 새로운 시각으로 병행성 프로그래밍에 대한 접근을 하고자 한다. 여기에 실용적으로 많이 사용되고 발전되어온 자바에서 적용하고자 한다.

pi-calculus 에 기반한 pict 를 모델로 하여 pict 에서 지원하는 아주 명료한 연산자와 current object 를 도입함으로써 보다 표현력과 확장성, 검증성이 뛰어난 고 수준 자바 라이브러리를 설계하고자 한다.

2 장에서는 자바에서 병행성 프로그래밍의 기법을 알아보고 이의 문제점에 대하여 논한다. 3 장에서는 병행성 이론 (pi-calculus)에 대하여 소개하고 pict 의 간결한 연산구조와 방법론을 소개한다. 4 장에서는 3 장에서 소개한 pi-calculus 에 기반한 고급자바 라이브러리 설계에 대한 제안을 제시한다. 마지막 결론에서는 본 연구의 결론과 나서는 과제, 향후 방향에 대하여 논한다.

2. Java concurrent programming

프로그램이 대형화되어가고 하드웨어의 발전은 병행성 프로그램의 필요성을 더욱 더해가고 있는 실정이다. 병행성을 지원하는 언어적 특징과 자바에서 지원하는 언어적 특성을 알아보자.

2.1 언어적인 특성

프로그래밍 언어는 병행성을 지원하기 위해서는 다음과 같은 요건을 갖추어야 한다.

첫째. 다중 프로세스 프로그램구조를 허용해야 한다.
둘째. 공유자원에 대한 접근제어를 표현해야 한다.
셋째. 프로그램 단위들을 동기화 하는 기능을 가져야 한다.

이러한 요건에서 프로그램들이 동시 수행될 가능성을 표현할 수 있게 되는 것이다. 또한 이러한 기능은 병렬 시스템에서 수행할 경우 아주 효율적인 수행능력을 나타낼 수 있다.

이러한 병행성은 자바에서도 언어적으로 지원하고 있다. 자바에서는 몇가지 핵심적인 특징으로 이러한 언어적 지원을 가능케 한다.

첫째. 동시수행을 위한 Runable 인터페이스.
둘째. Thread 클래스 지원
셋째. 공유자원 접근제어를 위한 Synchronized 예약어.
넷째. Object 클래스의 wait(), notify(), notifyAll() 메서드
다섯째. 모니터 메커니즘.

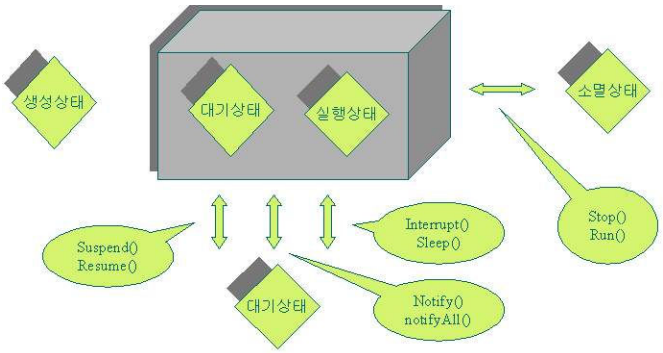

그림 1. 쓰레드 생명주기

자바에서는 쓰레드간의 정보 교환 수단으로 공유 기억장소를 사용한다. 하나의 쓰레드에서 구현된 기법은 여러 개의 쓰레드에서도 오류를 범하지 않고 실행할 할 수 있게 표현해야 한다. 모니터 메커니즘과 object 클래스의 메서드들은 구현에 중점을 둔 저수준으로 설계되어있다.

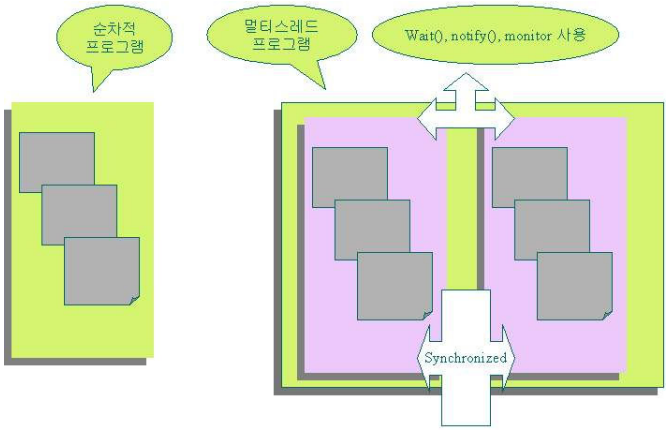

그림 2. 멀티스레드 프로그래밍 예

멀티 쓰레드를 구현하는데 있어서 Synchonized 예약어는 상호배제 원칙하에 공유자원에 대한 접근제어를 위해서 사용할 수 있다. 또한 위 그림 2 에서 조건변수들을 사용하는 쓰레드들은 모니터라는 메커니즘을 사용하여 동기화 시킨다.

2.2 자바 병행성의 문제

자바에서는 병행성을 지원하는데 있어서 여러 문제에서 빈번히 나타나는 제어패턴을 표현하기 위한 기능이기 보다는 이러한 패턴들을 구현할 수 있는 저수준으로 설계되었다. 이는 여러 가지 문제를 야기할 수 있다. 먼저 설계적 측면에서 살펴보자.

첫째로 문제해결을 위한 알고리즘과 병행성을 제어하는 구문이 복잡하게 연관되었을 때는 저수준으로 설계되어 있는 자바에서는 이해도를 저해 시킬 수 있다. 두 번째로는 자바에서 지원하는 동기화 기능은 단순한 형태의 동기화를 요구하는 프로그램에서 유용하게 사용할 수 있게 설계되어 있다. 해서 대형프로그램에서는 문제가 나타날 수 있다. 마지막으로 앞에서 언급한 문제 해결을 위하여 패턴표현에 대한 선행연구가 있지만 적용과정에서 의도하지 않은 오류가 발생할 가능성이 있다.

실제적 측면에서도 문제점을 가진다. 먼저 모니터모델에 대한 문제점이다 모니터모델은 상호배제 표현에 효과적인 메커니즘이다. 하지만 협동적인 관계를 적용하면 프로그래밍이 불편하게 된다. 이는 비직관적인 프로그램을 유도할 가능성이 생기게 된다. 둘째로는 경쟁관계 조정을 위해서 synchronized 선언된 메서드들이 상호교차 호출될 때 deadlock 을 일으킬 수 있다.

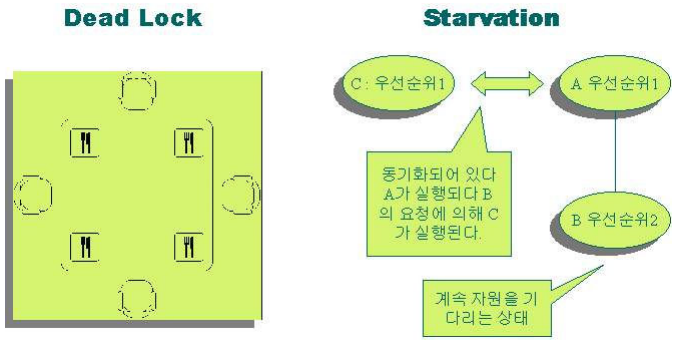

그림 3. 자바의 문제점

실제 자바에서 제공하는 쓰레드와 동기화 기능은 애플릿 같은 단순한 형태의 프로그램에서는 유용하지만 복잡한 병렬알고리즘의 구현을 위한 수단으로는 극히 원시적인 기능이라 할 수 있다.

하여 보다 표현적으로 직관적이고 명료한 설계가 필요하게 되고 이는 pi-calculus 의 병행성을 이용하면 확장할 수 있다. 다음 장에서 병행성 이론에 대하여 다루고자 한다.

3.병행성 이론 (pi-calculus)

pi-calculus 는 concurrent 나 distributed

programming 패러다임의 정수라고 할 수 있을 만큼 직관적이고 명료하다. 이런 pi-calculus 는 프로세스로 모든 expression 가능하게 한다. pi-calculus 의 몇 가지 특징을 살펴보면 다른 프로세스와 병렬로 실행하는 구조이다. 여기서는 많은 독립적인 서브 프로세스를 포함할 수가 있다. pi-calculus 는 기본적인 기법은 하나의 채널 위에서만 메시지를 교환하고 오직 channel 을 통해서만 가능한 처리기법을 사용하고 있다.

여기서 본 연구에서는 pi-calculus 에 기반한 pict 에 대하여 살펴 볼 필요가 있다. pict 는 pi-calculus 의 이론을 그대로 표현하고 있고 간단하지만 명시적인 연산자로 표현성을 확장하고 있다. 본 연구에서는 pict 가 구현하는 방식을 자바에서 확장하고자 한다. 먼저 pict 에 개략적인 소개를 하고자 한다.

3.1 pict base on pi-calculus

pict 는 완전한 의미론을 기술하고 있고 효율적인 수행을 가능케 한다. 의미론은 core language 에서 의미론을 정의함으로 명료성을 가지고 있다. 기본적으로 하나의 프로세스로 작동하게 되어있으며 데이터타입은 tuples 와 records 로 구성되고 패턴 매칭으로 확장한다. 특이하게 + 연산자를 허용하지 않고 엄격한 타입 체킹을 한다. pi-calculus 와 마찬가지로 채널을 통해서 교신하며 오직 input 을 통해서만 응답하고 있다. 특징이라고 하면 비동기식의 output 을 가지고 있다는 것이다.

간단한 pict 언어의 특징을 그림으로 살펴보면 다음과 같다.

run(new x:^[] (run x? [] = print!" Test Pict! " | x! []))

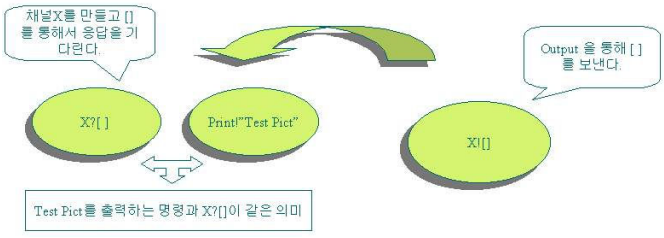

그림 4. pict 의 실행원리

간단한 원리이지만 정확한 rule 을 지키고 있다. 다음의 코드를 살펴보면 pict 의 특징을 모두 살펴볼 수 있다.

```
new b:^[^[] ^[]] new t:^[] new f:^[]
run b?[t f] = f![]
run (b![t f] | t? [] = print! "It.s true" | f?[] = print!"It's
false"))
new b: Boolean
run (ff! [ b ] | test! [ b ])
```

여기에서 우리는 채널만을 통한 교신원리와 타입 체킹, 기본적인 데이터 타입인 튜플과 레코드를 살펴볼 수 있다.

4. Java library for pi-calculus

pi-calculus 의 도입으로 자바의 기능을 확장할 수 있다. 이론적 도입과 함께 실제로 구현했을 때 다음의 고려사항이 나타날 수 있다.

4.1 Pi-calculus 의 제어추상화 이론도입

제어 추상화 이론을 도입함은 앞서 언급했듯이 자바에서의 문제해결방식으로 제어패턴을 표현하기 위한 기능이 아니라 패턴구현을 위한 저수준의 기능이기 때문이다. 자바에서 표현하지 못한 semantics 가 명료하게 됨으로써 분석, 검증을 정확하게 할 수 있다.

자바에서는 병행적 객체 설계구현의 어려움이 많이 나타난다. 하지만 이러한 병행적 객체는 pi-calculus 에서 정확하게 제시할 수 있다. 병행성은 객체지향 언어인 자바에서도 무엇보다 중요하다. 이렇게 도입된 제어추상화는 정확성에서 ambiguous 한 부분을 만들지 않음으로 프로그램의 정확성 증명하는데 유용하고 나아가 프로그램이나 알고리즘의 표현력과 확장성을 가져올 수 있을 것이다.

4.2 pi-calculus operator 도입

실제 구현에서 pict 에서 지원하는 연산자를 설계, 구현함으로써 자바의 병행성 기능을 확장하고자 한다. pict 관점에서 살펴보면 다음과 같다.

2.1 Structural congurrence

- Processes 는 α-convertible 이다.
- $(N/\equiv, +, 0)$ 은 symmetric monoid 이다.
- $N ::= \pi.P \mid 0 \mid M + N$ 이다.
- $(P/\equiv, +, 0)$ 은 symmetric monoid 이다.
- $!P \equiv P|!P$ 이다.
- $(vx)0 \equiv 0$, $(vx)(vy) \equiv (vy)(vx)P$ 이다.
- if $x \notin \mathrm{fn}(P)$ then $(vx)(P|Q) \equiv P|(vx)Q$ 이다.

2.2 reduction

- COMM : (… + x(y).P) | (…+ $\bar{x}$ z.Q) -> P{z/y}|Q
- PAR: $\dfrac{P \to P'}{P \mid Q \to P' \mid Q}$
- RES: $\dfrac{P \to P'}{(vx) \mid P \to (vx) \mid P'}$
- STRUCT : $\dfrac{Q \equiv PP \to P'\,P' \equiv Q'}{Q \to Q'}$

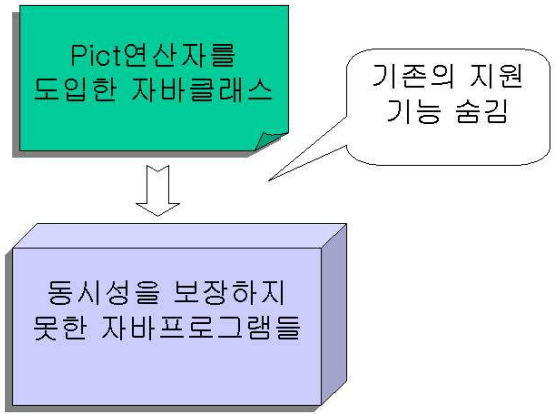

그림 5. 실제 구현원리

이러한 연산자를 실제로 도입하기 위해서는 기존의 자바의 병행성을 지원하는 특성과 이분화 또는 분리하여 적용하는 방법이 필요하다.

4.3. pi-calculus current object 도입

pi-calculus 에서는 Current object 를 이용하여 기능과 표현을 확장 시킨다. 다음에서 우리는 current object 를 볼 수 가 있다.

```
new contents:^Int
run contents!0
def set [v:Int c:Sig] =
      contents?_=
      ( contents!v | c![])
def get [res:/Int] =
      ( contents!v | res!v)

run ((prNL (int.toString (get)));
        (set 5);
        (prNL (int.toString (get)));
        (set -3);
```

이를 자바 라이브러리에 실용적으로 적용시켰을 때 이전의 비직관적인 표현수단의 기능을 향상시킬 수 있을 것으로 기대된다.

5. 결론

본 연구는 기존의 자바의 병행성 프로그램의 비직관적인 문제점을 Pi-calculus 병행적 이론의 도입하여 고수준의 제어 추상화와 자바기능을 확장하려 하였다. 또한 이렇게 확장된 고급 자바 라이브러리는 고수준 병행적 알고리즘 표현도구로서도 적용될 수 있을 것으로 기대된다. 이러한 라이브러리는 보다 병행, 병렬 프로그래밍을 쉽게 프로그래머에게 설계작성 할 수 있게 제공할 수 있고 이해도를 높일 수 있을 것이다. 단순한 표현의 확장성 뿐만이 아닌 의미론의 확대도 기대된다.
앞으로 연구과제는 실제구현에서 기존의 자바속성과의 제어패턴을 잘 조절해야 하고, 표현력과 실제 구현의 일치성이 요구된다.

참고문헌

[1] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 1977.

[2] Robin Milner. *Communication and Concurrency*. Series in Computer Science. Prentice-Hall Tnternational, 1989.

[3] Benjamin C. Pierce and David N. Turner. Concurrent object in a process calculus. *Theory and Practice of Parallel Programming (TPPP).* Number 907 in Lecture Notes in Computer Science, April 1995.

[4] David N. Turner. *The Polymorphic Pi-calculus: Theory and Implementation*. Ph. D. University of Edinburgh, 1995

[5] James Gosling, Bill Joy, and Guy Steele, *The Java Language Specification*, Addison-Wesley, 1996.

[6] Doug Lea, *concurrent Programming in Java* : Design Principles and Patterns, Addison-Wesley, 1997.

[7] B.P. Lester, *The Art of Parallel Programming*, Prentice-Hall, 1985.

[8] C.D. Marlin, Coroutines: *A Programming Methodology, a Language Design, and an Implementation*, Springer-Verlag, Berlin, 1980.

[9] S. Baase, *Computer Algorithms: Introduction to Design and Analysis,* 2nd edition, Addison-Wesley, Reading Mass, 1988.

[10] M.H Kim, “A Generalized Enumeration Mechanism for Java.” *Java Developer’s journal*, 3(5), 9-16, SYS-CON Publications, May 1998.

[11] van Glabbeek, R. and Goltz, U., *Equivalence notions for concurrent systems and refinement of actions*, Proc, 4th Conference on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Springer Verlag, Vol 379, pp237-248, 1988.

[12] Milner, R., *Calculi for synchrony and asynchrony*, Journal of Theoretical Computer Science Vol 25, pp267-310, 1983.