## 끝내면서

본문에도 썼지만, 나는 통계를 포함한 수학 전반에 대해 잘 알지 못한다. 그리고 컨커런트 프로그래밍도 잘 못한다. 그러나 빅 데이터와 데이터 과학 등의 말이 유행하고 컴퓨터의 멀티 코어 화가 진행되는 현대에서 내가 싫어하는 그 분야의 중요성은 계속 증가하고 있다.

우리 (대부분)는 엔지니어다. 문제가 있으면 기술로 그것을 해결하려는 것이 엔지니어의 정신이 아닐까 생각한다. 그래서 이러한 문제에 대한 내 나름의 해답이 Streem이었다. 무엇보다, 영감은 있어도 실력이 수반하지 않는 영역에서의 싸움은 어렵기 그지 없었다. 이 책에서도 여러 번 참회하고 있는대로, 실력이 미치지 못한 버그를 잡을 수 없었던 것과, 의욕과 시간이 부족해 완성도가 낮은 채로 남겨져있는 곳이 많이 있다. 그러나 그 ‘단점’은 Streem라는 언어 자체의 가치를 떨어뜨리는 것은 아니라고 생각한다.

사실, 나는 이 Streem의 실행 모델이 상당히 마음에 들어, 2015 년 RubyKaigi과 RubyConf의 키 노트에서는 Ruby의 미래 버전 (Ruby3)에 Streem의 컨커런시 모델을 기반으로 한 기능을 넣으려는 제안을 했었다. 진짜 그렇게 되기를 바랬지만, 최종 검토단계에서 Ruby3에는 사사다 고이치(笹田耕一)가 제안한 Guild (길드)라는 모델이 실제로 채용되었다. 이쪽이 표현력과 호환성이 뛰어나다는 판단이다.

Streem 의 추상도가 높은 컨커런시 모델은 세심한 처리를 할 수 없다는 점이 장점이자 단점이기도 하지만 Ruby3에 들어가는 것에서는 단점 부분이 무시될 수 없었다.

글쎄, 내 생각한 Streem 모델을 가져 오는 방식은, Ruby와 Streem의 다른 실행 모델을 하나로 하기 위해 ‘하나의 언어에 두 모델’ 이 존재하는, 다른 말로하면 ‘유사 한 두 언어가 섞인’ 모델이 되기 때문에 사용자에게 혼란이 커질 위험이 있었다. 채용되지 못한 것은 어쩔 수없는 일이라고 생각한다.

#### 못한 부분

본문 중에도 참회 한 것처럼, Streem은 아직 미완성이다. 특히 가비지 컬렉션 (GC)이 제대로 구현하지 못하는 것은 큰 문제다. 현재 Streem 처리 시스템은 대부분 메모리를 해제하지 않는다. 따라서 대량의 데이터를 처리하면 필연적으로 메모리를 탕진하게 된다.

4-4절에서는, 쓰레드마다 다른 공간에 객체를 할당하여 GC도 쓰레드마다 실시하는 것으로 효율을 올릴 것으로 설명했지만 실제로 구현하다 보니 예상보다 어려움이 있었다. 파이프 라인에서 데이터를 전달하면 그 개체는 다음 스레드에 넘겨지게 된다.또한 이 객체가 배열과 같은 다른 개체에 대한 참조를 포함하면, 거기에서 참조 된 객체를 재귀적으로 다음 스레드의 메모리 공간에 복사해야 한다. 이것이 의외로 귀찮아, 좀 미루고 말았다.

실용적인 처리 시스템을 목표로 하기에, 이 점은 어떻게든 해결하고 싶다고 생각하고 있다. 첫 번째 단계는 NaNBoxing라는 객체 표현을 변경 한 후, libgc를 이용하려고 생각하고 있다.

libgc는 링크를 거는 것으로 C 언어로 작성된 애플리케이션에 GC를 추가하는 라이브러리다. 그러나 ‘포인터의 값을 가공하지 말라’라는 제약 조건이 있고, 현재의 Streem이 채용하고있는 NaNBoxing에서는 이 제약 조건을 만족하지 못한다.

현대의 거의 모든 컴퓨터가 채용하고있는 부동 소수점 표현인 IEEE754에서는, 배정도 부동소수점수 64비트 중 부호부 1 비트, 가수부에 11 비트, 가수부에 52 비트를 사용하고 있다.

또한 계산 결과가 정의되지 않은 경우의 연산결과 (예를 들어 0으로 나누는 것 등)를 나타내는 값으로 NaN (Not a Number) 값이 정의되어 있다. ‘지수부가 모두 1’의 값도 NaN 값이라고 정해져 있다. NaN을 나타내는 값 하나만 있으면 좋은데, 실제로는 지수가 모두 1이면 모든 NaN입니다. 따라서 잘하면 NaN으로 해석되는 값의 범위가 52 비트 분의 정수 값을 포함시킬 수 있다. 이것이 NaN Boxing이다. 실제로 52 비트 중 4 비트 값의 종류를 나타내는 태그에, 나머지 48 비트를 실제 값을 저장하는 데 사용한다.

48 비트 밖에 없으면 64 비트 OS에서의 포인터를 포함 할 수없을것 같다. 그러나 실제로는 일부 예외를 제외하고 Linux를 비롯한 많은 OS에서 64 비트 시스템에서는 포인터 값으로 48 비트 밖에 사용하지 않기 때문에 문제가 될 수는 없다. 실로 교묘하다.

한편,이 방식에서는 포인터의 값이 부동소수점수로 변환되어 버리므로, libgc는 포인터를 찾을 수 없고, GC도 할 수 없다. 그래서 NaN Boxing 부분에 손을 대어 포인터 값을 포인터 그대로 유지하는 형식으로 변경한다.

발상은 간단하다. 64비트 포인터 값 중 실제로 주소를 표현하는것은 48 비트이고 나머지 16 비트는 0이다. NaN Boxing에서는 이 부분에 태그를 넣어 NaN 값으로 꾸며준다. 이렇게 하는 것은, 포인터 값을 표현하는 경우에 태그가 0이 되도록 태그 값을 조정해주면 좋을 것이다. 물론이 조정한 값은 정당한 부동 소수점수가 될 수 없다. 따라서 부동 소수점 숫자를 추출할 때 값을 가공 할 필요가 있지만, 그다지 큰 수고가 드는 것은 아니다.

이러한 NaN Boxing의 변종을 Favor Pointers라고 한다.

NaN Boxing을 Favor Pointers로 바꾸면 그 뒤는 간단해진다. 현재 malloc ()를 부르고 메모리를 할당하는 부분을 GC\_malloc ()로의 대체만으로 GC가 가능해진다. libgc는 4-4절에서 해설 한 것 같은 어플리케이션 고유의 지식을 이용한 효율화는 불가능하지만, 쓰레드에 대응하는 세대별 GC를 제공하고 있기 때문에 나름대로 효율적인 GC를 해 준다.

#### 안녕, 그리고 이제부터

Ruby3에 포함되지 않는다고 해서, Streem의 가치가 없어지는 것은 아니다. Streem는 독립적인 언어로서 앞으로도 (가늘고 길게) 개발을 계속하려고 생각하고 있다.

Streem에 가까운 대상 영역을 가진 언어 도구는 Streem외에도 존재한다. 예를 들어, tab[^1] 과 datamash[^2]이다. 이러한 언어 도구의 라이벌이 될 수 있도록 앞으로도 Streem를 성장 시키려고 한다.

마츠모토 선생의 다음 작품을 기대해 주세요!!

2016년 12월

마츠모토 유키히로

[^1]: http://tkatchev.bitbucket.org/tab/

[^2]: http://www.gnu.org/software/datamash

저자 : 마츠모토 유키히로(まつもとゆきひろ)

1965년생. 돗토리현 요나고시 출신. 츠쿠바대학 제3학군정보학류(筑波大学第三学群情報学類)졸업. 프로그래밍 언어 Ruby의 창시자. 프로그래밍 언어 디자인 제1인자. (아마도) 일본 유일의 프로그래밍 언어 디자이너. 주식회사 네트워크 응용통신연구소 특별연구원. 일반재단법인 Ruby 어소시에이션 이사장. Heroku Chief Architect등 직함 다수. 1남3녀와 개와 고양이 각 1마리의 아버지이기도 함. 온천을 좋아하고 시마네현에 거주중. 목양좌, O형.