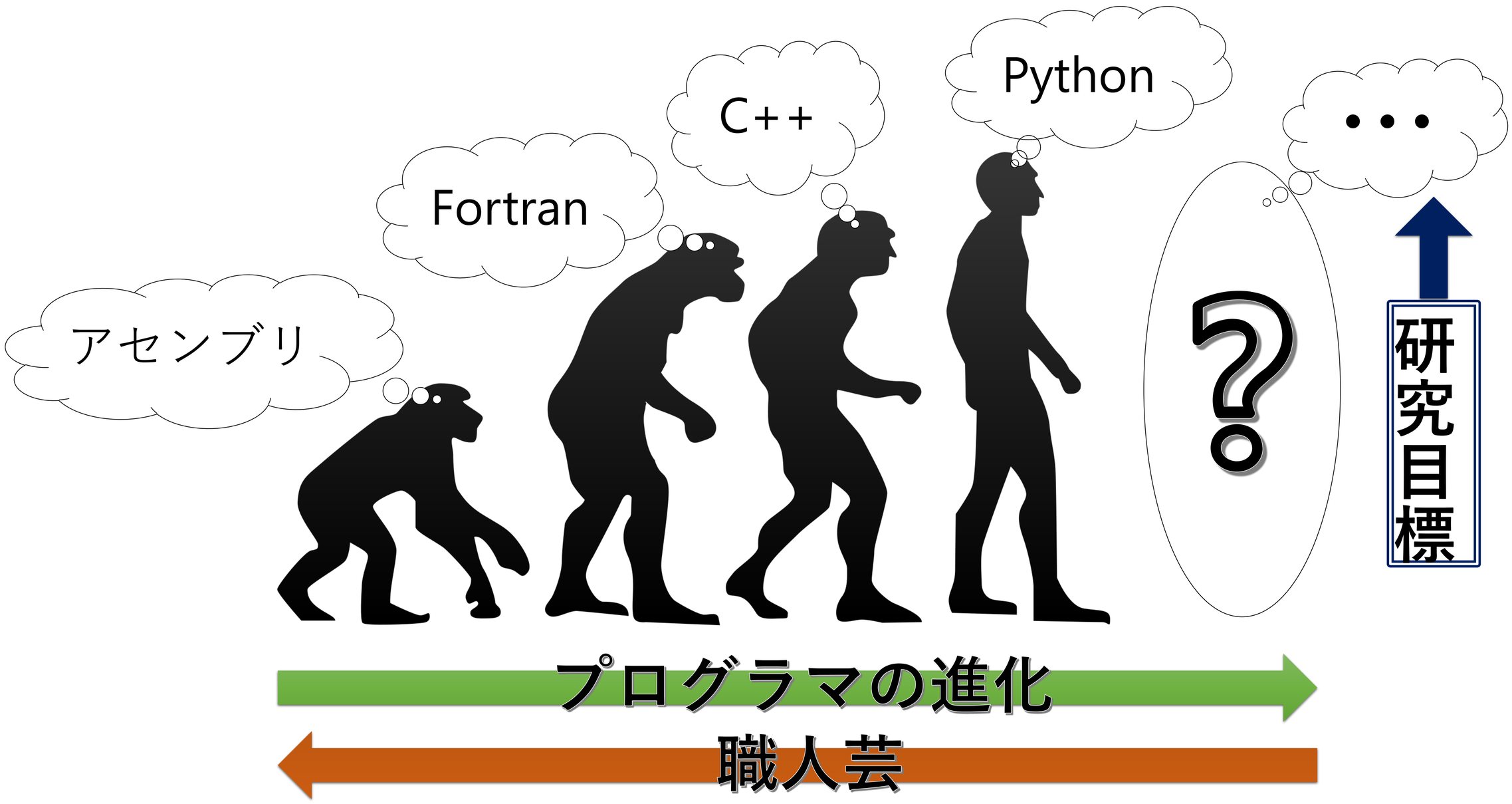
自動チューニング可能な
一般化N体問題解法枠組みの開発

佐藤重幸
（東京大学）

—職人芸のプログラミングを自動化—



職人芸
- プログラムの効率化に効果絶大
- プログラマを先祖返りさせる

先祖返りすると…
- ヒトが読めないプログラム
- 融通が利かないプログラム

負債

職人芸の自動化 ⇒
- ✔**負債化しない**高性能プログラム
- ✔プログラマの**進化を促進**

研究成果
- N体問題という問題領域に特化した言語の設計
- 職人芸を自動化する言語処理系基盤の開発

```
gravity(AstroParticle<double,D>> *particles, int n) {
  assert(n % 4 == 0);
  for (int i = 0; i + 3 < n; i += 4) {
    const auto base_ptr = reinterpret_cast<double*>(particles + i);
    vcl::Vec4q m_idx(&particles[i  ].m - base_ptr, &particles[i+1].m - base_ptr,
                     &particles[i+2].m - base_ptr, &particles[i+3].m - base_ptr);
    auto m = vcl::lookup<sizeof(AstroParticle<double,3>) / sizeof(double) * 4>(m_idx, base_ptr);
    vcl::Vec4d pos[D], acc[D];
    for (int k = 0; k < D; k++) {
      vcl::Vec4q pos_idx(&particles[i  ].x[k] - base_ptr, &particles[i+1].x[k] - base_ptr,
                         &particles[i+2].x[k] - base_ptr, &particles[i+3].x[k] - base_ptr);
      vcl::Vec4q acc_idx(&particles[i  ].a[k] - base_ptr, &particles[i+1].a[k] - base_ptr,
                         &particles[i+2].a[k] - base_ptr, &particles[i+3].a[k] - base_ptr);
      pos[k] = vcl::lookup<sizeof(AstroParticle<double,D>) / sizeof(double) * 4>(pos_idx, base_ptr);
      acc[k] = vcl::lookup<sizeof(AstroParticle<double,D>) / sizeof(double) * 4>(acc_idx, base_ptr);
    }
    vcl::Vec4q i4(i, i+1, i+2, i+3);
    for (int j = i; j < n; j++) {
      vcl::Vec4d r[D];
      vcl::Vec4d r2 = 0;
      for (int k = 0; k < D; k++) {
        auto tmp = vcl::Vec4d(particles[j].x[k]) - pos[k];
        r[k] = tmp;
        r2 += tmp * tmp;
      }
      const auto r3 = r2 * vcl::sqrt(r2);
      auto coef = PhysicalConstant<double>::G / r3;
      coef = coef & vcl::Vec4db(i4 != j);
      auto coef_i = coef * particles[j].m;
      auto coef_j = coef * m
      for (int k = 0; k < D; k++) {
        acc[k] += coef_i * r[k];
        particles[j].a[k] += vcl::horizontal_add(coef_j * -r[k]);
      }
    }
    for (int ii = 0; ii < 4; ii++) {
      for (int k = 0; k < D; k++) {
        particles[i+ii].a[k] = acc[k][ii];
      }
    }
  }
}
```

$$\vec{a}_{12} = \frac{Gm_1}{\|\vec{x}_1 - \vec{x}_2\|^3}(\vec{x}_1 - \vec{x}_2)$$

```
@mutualize
def gravity(p1, p2):
  r = [0.0 for _ in range(D)]
  r2 = 0.0
  for i in range(D):
    r[i] = p1.x[i] - p2.x[i]
    r2 += r[i] * r[i]
  r3 = r2 * sqrt(r2)
  for i in range(D):
    da = G * p1.m / r3 * r[i]
    p2.a[i] += p2.a[i] + da
```