C++ Optimizations

Vipin Vasu

Outline

Temporaries

```
class vec3d {
  double x,y,z;
  friend vec3d operator*(double, const vec3d&);
public:
  vec3d(double _x=0.0, double _y=0.0, double _z=0.0) : // 4 ctors
        x(_x),y(_y),z(_z) {}
  vec3d(const vec3d &other);
  vec3d operator=(const vec3d &other);
  vec3d operator+(const vec3d &other) {
    vec3d tmp;
    tmp.x = x + other.x;
    tmp.y = y + other.y;
    tmp.z = z + other.z;
  }
  vec3d operator*(const vec3d &other);
  ...
};

vec3d operator*(double s, const vec3d& v) {
  vec3d tmp(s*v.x,s*v,y,s*v.z);
}
```

Lazy Construction

```
void f(double threshold,int length)
{
    std::vector v(length);
    if(rand() > threshold*RAND_MAX)
    {
        v = obtain_data(length);
        std::sort(v.begin(), v.end());
        process_data(v);
    }
}
```

Lazy Construction

```
void f(double threshold, int length)
{
    if(rand() > threshold*RAND_MAX)
    {
        std::vector v(obtain_data(length));
        std::sort(v.begin(), v.end());
        process_data(v);
    }
}
```

Static Construction

```
const int length=1000;
void f(double threshold)
{
    static std::vector v(length);
    if(rand() > threshold*RAND_MAX)
    {
        v = obtain_data(length);
        std::sort(v.begin(), v.end());
        process_data(v);
    }
}
```

Loop Kernels and Iterators

```
template T sprod(const vector &a, const vector &b)
{
    T result=T(0);
    int s = a.size();
    for( int i = 0; i < s; ++i )
        result += a[i] * b[i];
    return result;
}
```

Loop Kernels and Iterators

```
template T sprod(const vector &a, const vector &b)
{
    typename vector::const_iterator ia=a.begin(),
    ib=b.begin();
    T result=T(0);
    int s = a.size();
    for(int i=0; i<s; ++i)
    result += ia[i] * ib[i];
    return result;
}
```

The End