Activity analysis for reverse-mode differentiation of (CUDA) GPU kernels

Mentors: Vassil Vassilev, David Lange

19.06.2025

Removing exessive AtomicAdd's

```
__global__ void kernel_call(double *out, double *in) {
  int index = threadIdx.x + blockIdx.x * blockDim.x;
  out[index] = in[index];
}

void fn(double *out, double *in) {
  kernel_call<<<1, 10>>>(out, in);
}
```

```
{
    out[index0] = _t2;
    double _r_d0 = _d_out[index0];
    _d_out[index0] = 0.;
    atomicAdd(&_d_in[index], _r_d0);
}
```

```
_d_in[index] += _rd0;
```

Removing exessive AtomicAdd's

```
__global__ void kernel_call(double *out, double *in) {
  int index = threadIdx.x;
  out[index] = in[index];
}

void fn(double *out, double *in) {
  kernel_call<<<1, 10>>>(out, in);
}
```

Loop analysis and optimizations

- Sparsity Patterns

```
double f1(double a[100], double b double _clad_out_output[]){
  for(int i = 0; i<100; i++)
    a[i] = i * b;
                          . . .
}
```

Loop analysis and optimizations

- Merging forward and reverse loops

```
void only_for(double* x, double y){
  for(int i = 0; i<100; i++)
    x[i] += y;
}
```

```
void only_for_pullback(…) {
        . . .
    for (i = 0; ; i++) {
        {
            if (!(i < 100))
                break;
        }
        _t0++;
        x[i] += y;
    }
    for (;; _t0--) {
        {
            if (!_t0)
                break;
        }
        i--;
        double _r_d0 = _d_x[i];
        *_d_y += _r_d0;
    }
}
```

```
void only_for_pullback(…) {
        . . .
    for (i = 0; ; i++) {
        {
            if (!(i < 100))
                break;
        }
        x[i] += y;
        {
            double _r_d0 = _d_x[i];
            *_d_y += _r_d0;
        }
    }
```

Reworking TBR and AA

```
struct VarData {
    enum VarDataType { UNDEFINED, FUND_TYPE, OBJ_TYPE, ARR_TYPE, REF_TYPE };
    union VarDataValue {
      bool m_FundData;
      std::unique_ptr<ArrMap> m_ArrData;
      Expr* m_RefData;
      VarDataValue() : m_ArrData(nullptr) {}
      ~VarDataValue() {}
    };
    VarDataType m_Type = UNDEFINED;
    VarDataValue m_Val;

                  . . .
  };
```

```
using VarsData = std::set<const clang::VarDecl*>;
```

suggestions?