Simplicity is Complicated: On the Balance of Performance and Knobs

Changkun Ou
LMU Munich

IDC 2019 Autumn
Vienna, Austria
Oct. 9, 2019
Changkun Ou (欧长坤)
Tschang-Kwën Ou
/ʈʂʰɑŋkuən ɤʊ /

- PhD student / Prof. Butz
- Expertise: Web Techs, Machine Learning, Distributed Systems

- Contributor of **Tensorflow, Go, etcd, redis, …** (100k+ ⭐️ projects)
- Author of C++ and Go books (6k+ ⭐️ projects)
- Many many other open source contributions...

https://changkun.de
Mesh simplification (MS)
Mesh simplification (MS)

MS is largely applied
Mesh simplification (MS)

MS is largely applied

MS has been "solved" 20 years ago [Hoppe'96] [Garland'97]
Blender

MeshLab

Autodesk Maya
2-Manifold locally resembles 2D Euclidean space.

// Polyreduce reduces number of polygons
// while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {
    for !m.Eval(c) {
        local := m.Pick()
        local.Simplify()
    }
}

Issues:

1. Non-scalable (serialized process)
2. NP-hard computation
3. ...
What really matters? Practitioner's Perspective

1. Reduction speed → Computation complexity
What really matters? Practitioner's Perspective

1. Reduction speed → Computation complexity
2. Expert preference → Reduction quality
3. Manual intervention → Automation level

Can you tell the difference?
1. Reduction speed → Computation complexity
2. Expert preference → Reduction quality
3. Manual intervention → Automation level

Can you tell the difference?
The Ultimate MS Program

Target #Poly: _______________

Run!
The Ultimate MS Program

1. Reduction speed → Computation complexity
   
   **Solution:** Thread multiplexing 🔥🔥🔥

2. Expert preference → Reduction quality
3. Manual intervention → Automation level
   
   **Solution:** Hyperparameter reduction & Imitation learning 😎😎😎

Impractical
Computation Complexity:
From Parallelism to Concurrency
// Polyreduce reduces number of polygons
// while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {
    for !m.Eval(c) {
        local := m.Pick()
        local.Simplify()
    }
}

// SGD implements mini-batch
// stochastic gradient descent.
func SGD(m *Model, d *Dataset) {
    for !m.Converge() {
        miniB := d.Batch()
        m.GradientDescent(miniB)
    }
}
Concurrent MS

// Polyreduce reduces number of polygons // while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {
    for !m.Eval(c) {
        local := m.Pick()
        local.Simplify()
    }
}

// SGD implements mini-batch // stochastic gradient descent.
func SGD(m *Model, d *Dataset) {
    for !m.Converge() {
        miniB := d.Batch()
        m.GradientDescent(miniB)
    }
}

Inspiration

// Polyreduce reduces number of polygons // while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {
    // SPEEDUP1: build workQ concurrently
    for local := m.Pick(); local != nil; {
        sched.Submit(func() {
            quality, ok := local.Eval(c)
            if ok {
                workQ.Push(quality, local)
            }
        })
        local = m.Pick()
    }
    sched.Wait() // sync barrier

    // SPEEDUP2: run workQ concurrently
    for w := workQ.Pop(); w != nil; {
        sched.Submit(w.Simplify)
        w = workQ.Pop()
    }
    sched.Wait() // sync barrier
}

var sched Sched // M:N work-steal scheduling
func (s *Sched) Submit(f func()) { ... }
Thread Multiplexing: Work-steal scheduling
We are not even close

640k#poly
Ground truth

29k#poly
Handcraft
40 hours+

119k#poly
My approach
~30s

126k#poly

[Hoppe96]+[Garland97]
in Blender and many other 3D softwares
~6min
We are not even close

Ground truth

Handcraft
40 hours+

640k#poly

29k#poly

119k#poly

My approach
~30s

126k#poly

[Hoppe96]+[Garland97]
in Blender and many other 3D softwares
~6min

"MS is a solved problem!"
Quality & Automation:
From Fully Automatic to Semi Automatic
\[
\sum_{\text{model}} \text{cost(model, method)} = \text{const.}
\]
Modeling Human Cost in MS

\[
T = \left[ MS + DT(p) \right] \times A(p)
\]

If \( p_1 > p_2 \):

\[
T_1 - T_2 = MS \times \left[ A(p_1) - A(p_2) \right] + \left[ DT(p_1) \times A(p_1) - DT(p_2) \times A(p_2) \right]
\]


\[ T = [ MS + DT(p) ] * A(p) \]

If \( p_1 > p_2 \):

\[ T_1 - T_2 = MS * [ A(p_1) - A(p_2) ] + [ DT(p_1) * A(p_1) - DT(p_2) * A(p_2) ] \]

\[ > MS * [ A(p_1) - A(p_2) ] + [ DT(p_2) * A(p_1) - DT(p_2) * A(p_2) ] \]

Hick's Law
Modeling Human Cost in MS

\[
T = [ \text{MS} + \text{DT}(p) ] \times A(p)
\]

If \( p_1 > p_2 \) :

\[
T_1 - T_2 = \text{MS} \times [ A(p_1) - A(p_2) ] + [ \text{DT}(p_1) \times A(p_1) - \text{DT}(p_2) \times A(p_2) ]
\]

\[
> \text{MS} \times [ A(p_1) - A(p_2) ] + [ \text{DT}(p_2) \times A(p_1) - \text{DT}(p_2) \times A(p_2) ]
\]

\[
= [ \text{MS} + \text{DT}(p_2) ] \times [ A(p_1) - A(p_2) ]
\]

\[\Rightarrow\text{Reduce attempts}\]
Two Groups of Web APIs (*talk to me for beta access*)

```go
type PolyReduce interface {
    Upload(m *Model) (OpID string)
    Upload(OpID string, c *Config)
    Run(OpID string) // 1m #poly ≈ 1 min → 1 model
    Download(OpID string) (m *Model)
}

type ProPolyReduce interface {
    Upload(m *Model) (SessID string)
    Run(SessID string) (OpIDs []OpID) // 1m #poly ≈ 2 min → 4 models
    Eval(OpIDs []OpID, Scores []int)
    Download(OpID string) (m *Model)
}
```
Automation, Speed, Quality Tradeoff

- Manual
- Automatic
- Low Comp. Complexity
- High Comp. Complexity

Obvious Stupid

Algorithmic Research

Impractical
Automation, Speed, Quality Tradeoff

- **Obvious Stupid**
- **Algorithmic Research**
- **Current Practice**
- **Impractical**

Axes:
- **High Comp. Complexity**
- **Automatic**
- **Manual**
- **Low Comp. Complexity**
Automation, Speed, Quality Tradeoff

- Obvious Stupid
- Algorithmic Research
- Current Practice
- Maybe Interesting
- Impractical
Bernhard Riemann (1826-1866)
"The performance improvement does not materialize from the air, it comes with code complexity increase."

Why MS is a well-studied problem to you? Or, why it isn't?

When did you start thinking about using MS? What are your expectations from MS? Why?

How did you evaluate MS outcomes in your 3D projects?

What are principles qualifying 3D Artists v.s. Non-3D Artists? Is it quantitative measurable? Why?

When did industrial MS fail to your case? Why?

What is your maximum tolerance of X to MS? Why? where X = speed, features preservation, and …

How could human solve retopology so incredible? What did we miss in MS?

Any thing in mind?