

---

Subject: Re: The age of multicore has just started for U++....

Posted by [mirek](#) on Sun, 08 Apr 2007 15:14:59 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

exolon wrote on Sun, 08 April 2007 10:33 What does this Cowork class do, and how?

Well, detailed description will come with new Core, but essentially it is a thread pool. A tool to simply paralellize loops.

E.g. this is some original loop from the uppweb:

```
VectorMap<String, String> reflink;
```

```
struct ScanTopicIterator : RichText::Iterator {  
    String    link;
```

```
    virtual bool operator()(int pos, const RichPara& para)  
    {  
        if(!IsNull(para.format.label)) {  
            reflink.Add(para.format.label, link);  
        }  
        return false;  
    }  
};
```

```
void GatherRefLinks(const char *upp)  
{  
    Progress pi;  
    pi.AlignText(ALIGN_LEFT);  
    for(FindFile pff(AppendFileName(upp, "*.tpp")); pff; pff.Next()) {  
        if(pff.IsFolder()) {  
            pi.Step();  
            String package = pff.GetName();  
            String pdir = AppendFileName(upp, package);  
            TopicLink tl;  
            tl.package = package;  
            for(FindFile ff(AppendFileName(pdir, "*.tpp")); ff; ff.Next()) {  
                if(ff.IsFolder()) {  
                    String group = GetFileTitle(ff.GetName());  
                    tl.group = group;  
                    String dir = AppendFileName(pdir, ff.GetName());  
                    for(FindFile ft(AppendFileName(dir, "*.tpp")); ft; ft.Next()) {  
                        if(ft.IsFile()) {  
                            String path = AppendFileName(dir, ft.GetName());  
                            tl.topic = GetFileTitle(ft.GetName());  
                            String link = TopicLinkString(tl);  
                            pi.SetText("Indexing topic " + tl.topic);  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

    ScanTopicIterator sti;
    sti.link = link;
    ParseQTF(ReadTopic(LoadFile(path))).Iterate(sti);
}
}
}
}
}
}
}
}
}
}

```

parallel version with CoWork:

```

StaticCriticalSection  reflink_lock;
VectorMap<String, String> reflink;

```

```

struct ScanTopicIterator : RichText::Iterator {
    String    link;

```

```

    virtual bool operator()(int pos, const RichPara& para)
    {
        if(!IsNull(para.format.label)) {
            INTERLOCKED_(reflink_lock)
            reflink.Add(para.format.label, link);
        }
        return false;
    }
};

```

```

static void sDoFile(const char *path, const char *link)
{
    ScanTopicIterator sti;
    sti.link = link;
    ParseQTF(ReadTopic(LoadFile(path))).Iterate(sti);
}

```

```

void GatherRefLinks(const char *upp)
{
    CoWork work;
    Progress pi;
    pi.AlignText(ALIGN_LEFT);
    for(FindFile pff(AppendFileName(upp, ".*")); pff; pff.Next()) {
        if(pff.IsFolder()) {
            pi.Step();
        }
    }
}

```

```
String package = pff.GetName();
String pdir = AppendFileName(upp, package);
TopicLink tl;
tl.package = package;
for(FindFile ff(AppendFileName(pdir, "*.tpp")); ff; ff.Next()) {
  if(ff.IsFolder()) {
    String group = GetFileTitle(ff.GetName());
    tl.group = group;
    String dir = AppendFileName(pdir, ff.GetName());
    for(FindFile ft(AppendFileName(dir, "*.tpp")); ft; ft.Next()) {
      if(ft.IsFile()) {
        String path = AppendFileName(dir, ft.GetName());
        tl.topic = GetFileTitle(ft.GetName());
        String link = TopicLinkString(tl);
        pi.SetText("Indexing topic " + tl.topic);
        work & callback2(sDoFile, path, link);
      }
    }
  }
}
}
```

The advantage is that you do not have start/join/manage threads, in fact, CoWork starts a couple of threads ("thread pool") first time it is used and never quits them until application exists.

Mirek

---