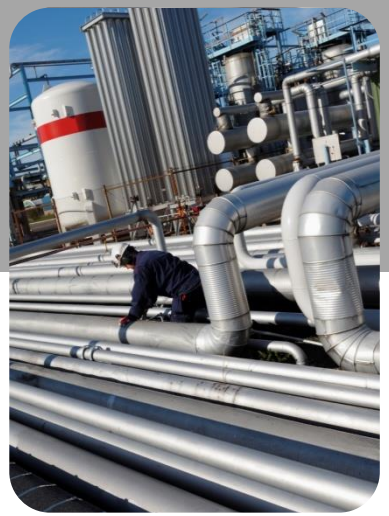
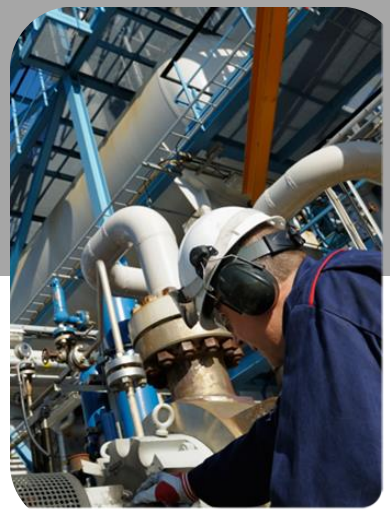
INFOCHEM

physical properties group of

KBC

Multiflash Thermo Package Developments: Support for Multithreaded CAPE-OPEN Clients

Victor Rühle, Francisco Martinez and Richard Szczepanski

CAPE-OPEN 2015 Annual Conference

13 October 2015

Outline

- Multiflash - introduction
- Motivation for change
- Software development
- Application examples
- Summary

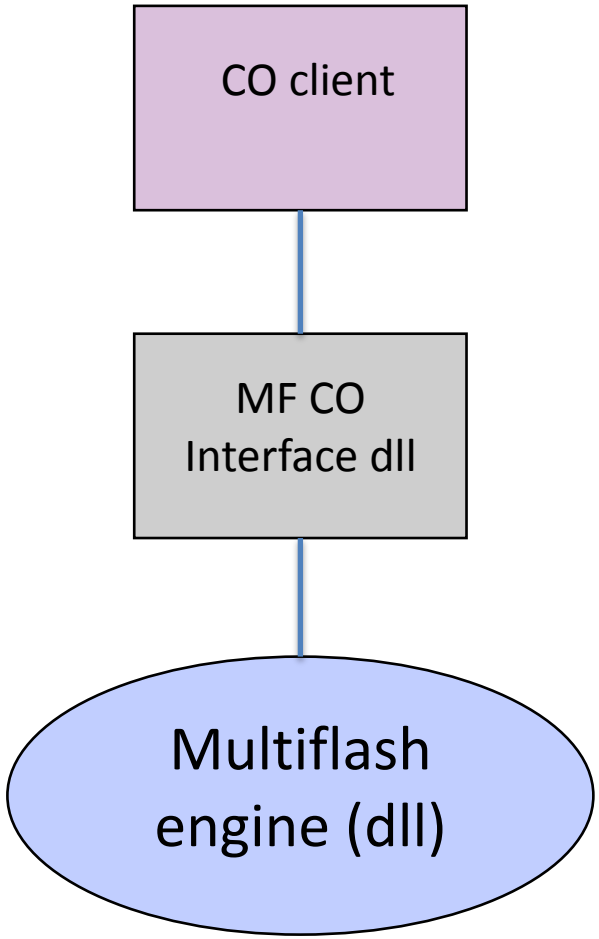
Multiflash: what it does

INFOCHEM
KBC

- Comprehensive PVT and physical properties package
  - Wide range of thermodynamic and transport property models for oil and gas, chemicals, refrigerants etc.
  - Phase equilibria for multiple fluid & solid phases
  - Fluid characterisation based on PVT lab data
- Main applications
  - Production/petroleum engineering
  - Flow assurance engineering
  - Process simulation
  - Equipment/plant design
  - On-line and real time simulation systems and flow-monitoring

Multiflash and CAPE-OPEN

- Supports CO Thermo Spec 1.0 and 1.1
- Includes Property Package Manager and Property Packages with persistence and editing support
- Uses Multiflash GUI to configure PP
- Full multiphase gas/liquids/solids capabilities in the CO 1.1 version
- Supports all (useful):
  - Phases
  - Properties
  - Derivatives
  - Flashes, etc.….



Background

- Multiflash original design 1988/9
  - Total development effort to-date ~ 90 man-years
  - Has proved to be flexible and extensible
  - Mainly Fortran code
- Evolution
  - Fluids
  - Hydrates and pure solids
  - Scales, waxes, asphaltenes
- Current position
  - Widely used in upstream industry with reputation for high quality and excellent support
  - Reliable
  - High performance

Motivation for Change

INFOCHEM
KBC

- Support large simulations
  - Computationally demanding applications, for example
    - Dynamic simulation of pipe networks in OLGA or LedaFlow
    - Large problems – e.g. multiple-refinery optimization in Petro-SIM™, Greater Plutonio life-of-field multi-field network simulation in Maximus™
    - Parametric studies
- Exploit the hardware
  - Multi-processor/multi-core hardware
  - 4, 16, …128 cores
- Current limitations
  - Static storage
    - limits some applications and leads to large memory requirement
    - Number of components, BIPs and phases
  - Cannot support multiple instances of the calculation routines
    - i.e. is not thread safe
    - Needs locks

Software Development

INFOCHEM

KBC

- Key features
  - Dynamic memory management
  - Thread safe
- Key objectives
  - Compatibility
  - Maintainability
  - Extensibility
  - Reliability
  - Performance

Software Development

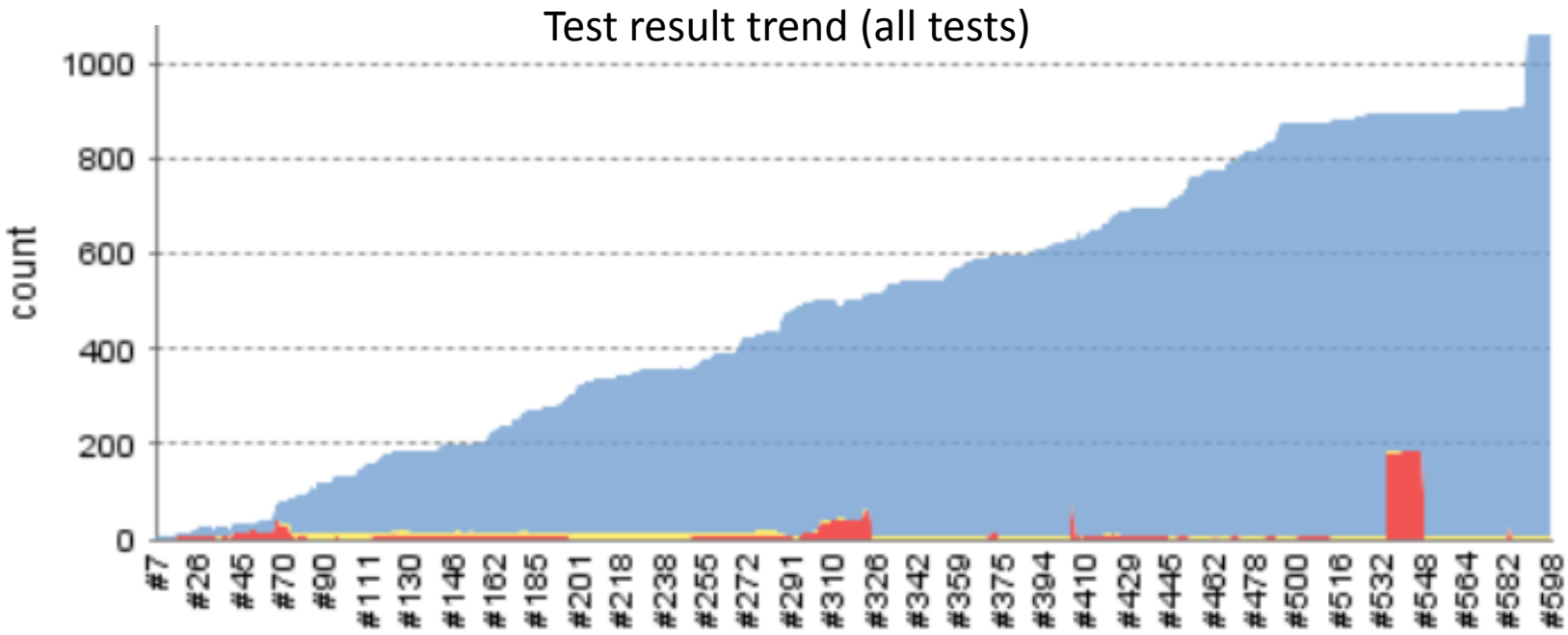
INFOCHEM
KBC

- Strategy
  - Redesign using object-oriented approach
  - Decoupled components and code re-use
  - Extensive testing
    - Unit
    - Integration
    - End-to-end
    - Regression
  - Continuous delivery
    - Nightly builds
    - Regular testing in Multiflash GUI
    - Fall-back to MF6 codebase for any features not yet implemented
  - Compatible wrapper interfaces for current API

Extensive testing during development

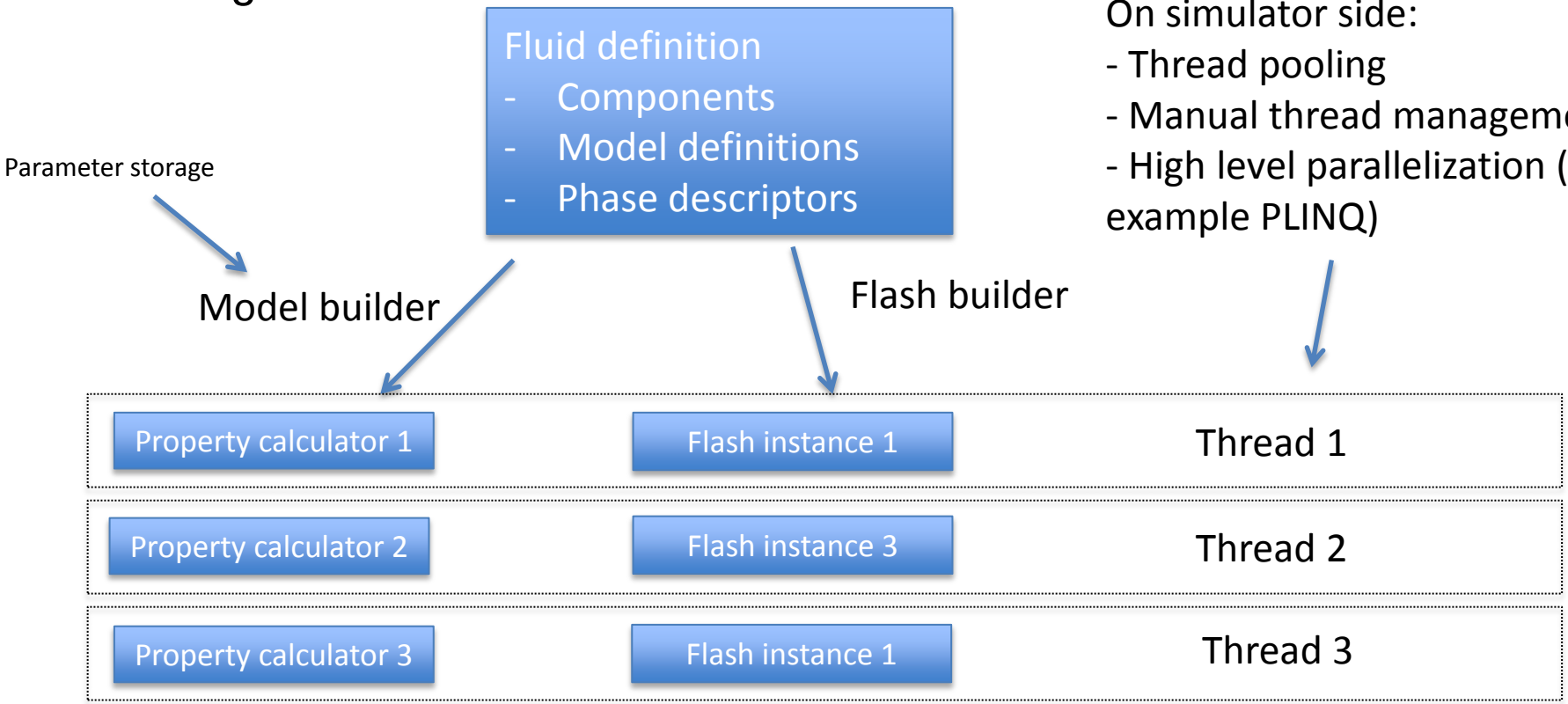
INFOCHEM KBC

- Testing frameworks
  - Googletest/googlemock
  - python unittest
- Jenkins build server
  - Nightly builds on Windows + Linux



MT concept

- No internal multithreading for flashes
  - Parallelization is responsibility of application, eg. the process simulator
  - Support easy creation of instances which can be distributed amongst threads



Example of API usage (work in progress)

INFOCHEM KBC

- API must be plain C
  - C++ APIs are compiler dependent (no standard application binary interface)
  - some compatibility within strong limitations
- Simple C api to allow for easy wrapping in higher level languages

Plain C API

```
bool success = mf_flash_create( stream, &flash );
if ( !success ) return false;

success = mf_ptflash_run( flash, 200, 1e6, feed, 1 );
if ( !success ) return false;

int numPhases = mf_flash_get_num_phases( flash );
mf_flash_get_phase( flash, 0, phaseDescriptor,
    phaseType, fraction, composition, ... );

mf_dispose( flash );
```

C# wrapper

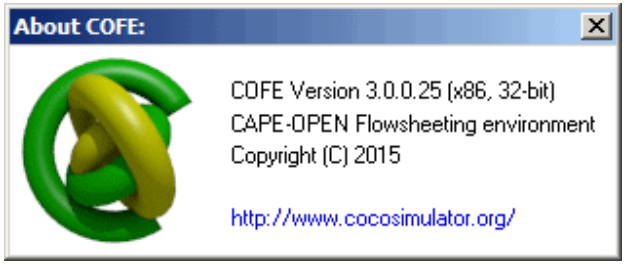
```
try {
    var flash = MultiflashWrapper.CreateFlash(stream);

    var result = flash.RunPTFlash(200, 1e6, feed);

    foreach (var phase in result.Phases) {
        var composition = phase.Composition;
        …
    }
}
catch (FlashFaileException e) { … }
```

CAPE-OPEN interface

- Current Multiflash
  - Support for multiple Property Packages (PP) requires locks and unloading/loading of model components
  - Total number of components, models, phases is limited
  - Version 6.0SP1 fixes issue with performance in single threaded mode and crash when running multithreaded
- Multiflash thread safe
  - Removed locks
  - No logical limit on number and size of PPs
  - Not feature-complete so fall-back to MF6
- Comments on the development process
  - Independent testing not straightforward
  - No readily available Material Object code?
  - COFE Version 3.0 used to test the Multiflash PP

Examples

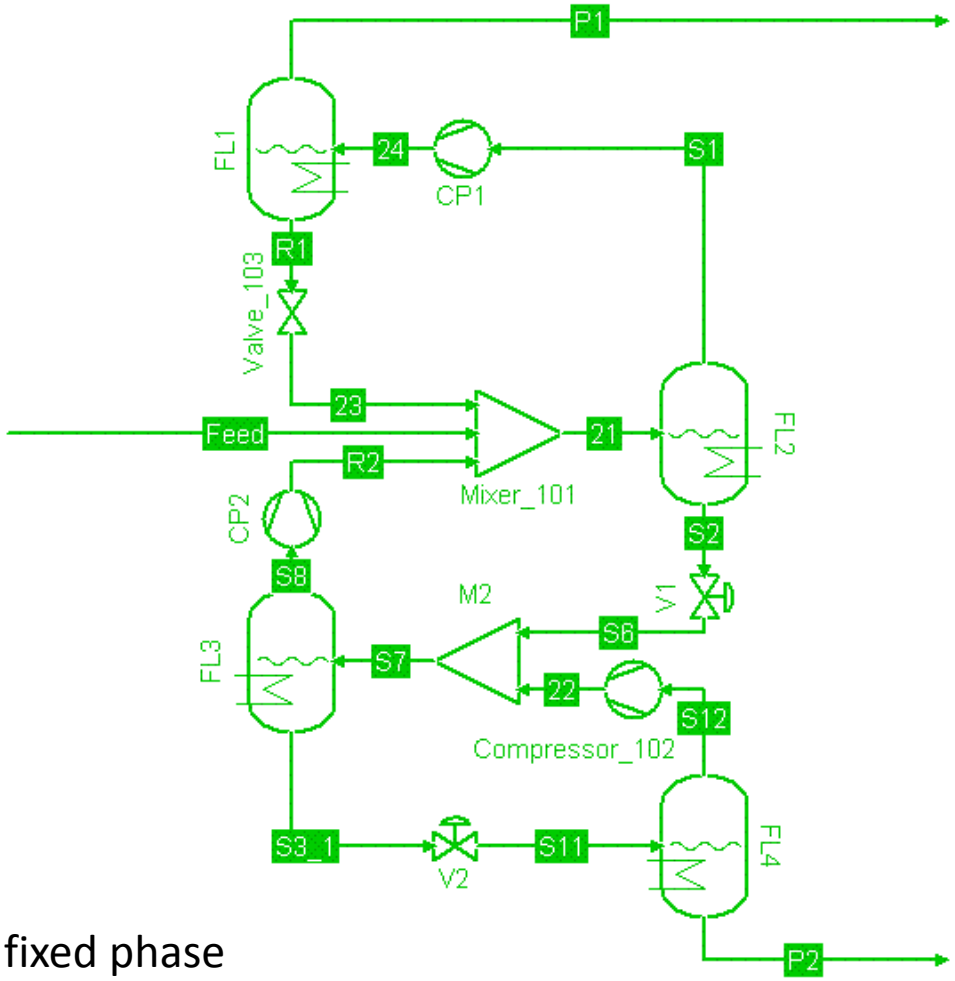
- Cavett process – COFE
- Refinery light-ends separation with parametric study – COFE
- Flash calculations for complex phase diagram including hydrates and ice
- Timings taken on Core i7-4770 3.4 GHz (4 cores)



Cavett Process - COFE

- 16 components
- PR eos
- Phases
  - Vapour
  - Liquid

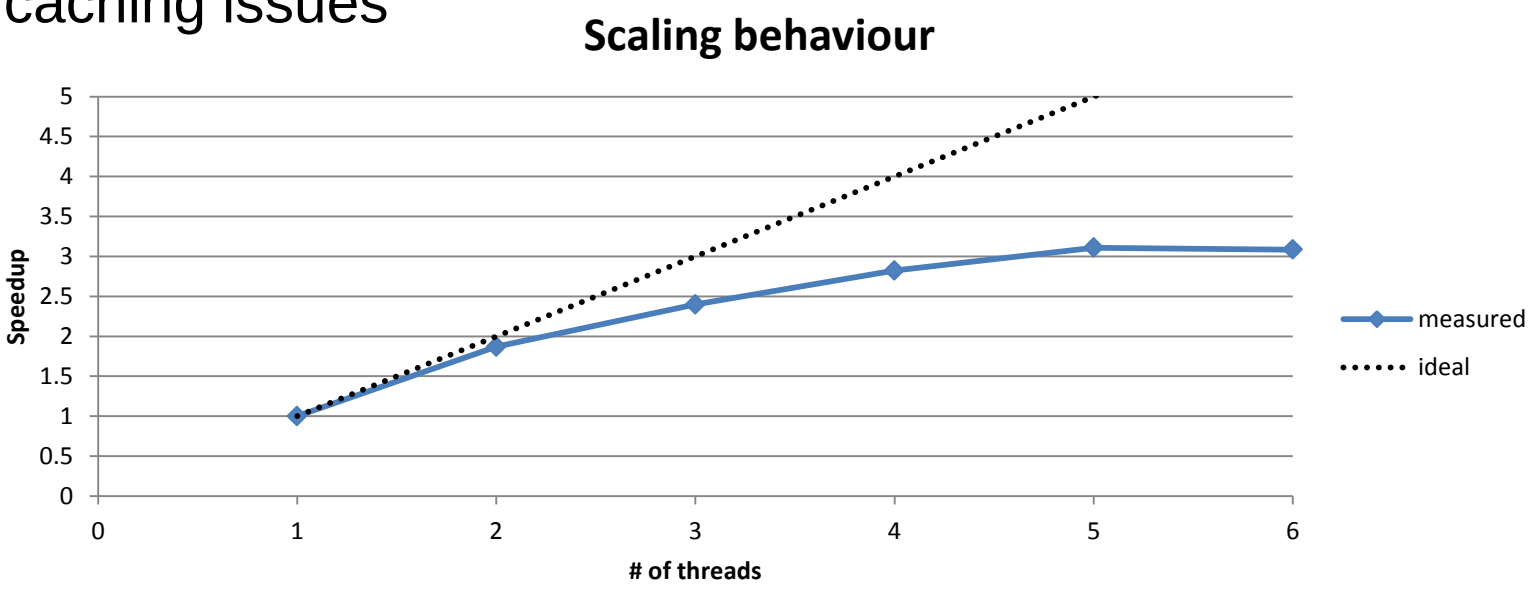
- Flow sheet solves in ~0.5s, hard to benchmark accurately
- Slight improvement with multiple threads
  - Cost of creating a material objects for each thread?



Bottleneck currently: ~50% time spent in fixed phase fraction flash (bubble point) calculations

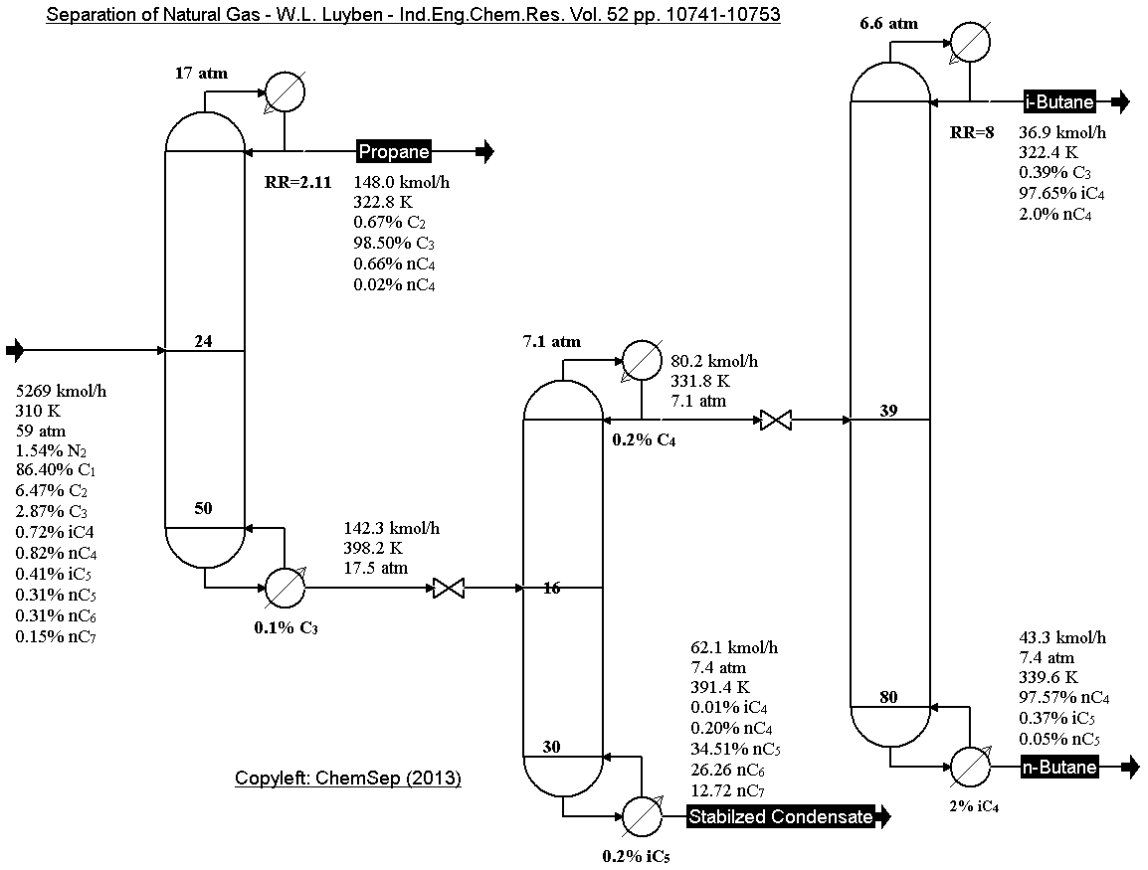
Cavett: Parametric study

- Parametric study
  - T, P in input feed
  - 30x30 points
- Possible reasons for non-ideal scaling
  - GUI updates, competing with other processes, multithreading implementation in COFE, global heap allocator lock, memory caching issues



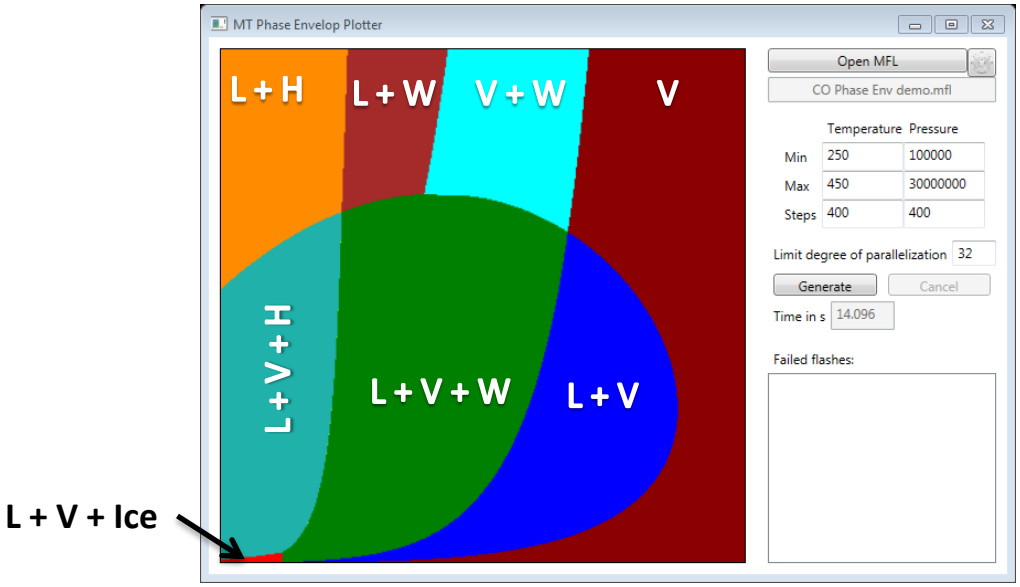
Refinery light-ends separation - COFE

- 10 components
- PR eos
- Phases
  - vapour
  - Liquid
- Solved in ~8s
  - Doesn't seem to be limited by thermo package
- Parametric study speedup 3.8



Phase diagram - Multiflash

- 10 components
- CPA eos
- Phases
  - Vapour
  - HC liquid
  - Water
  - Hydrate II
  - Ice

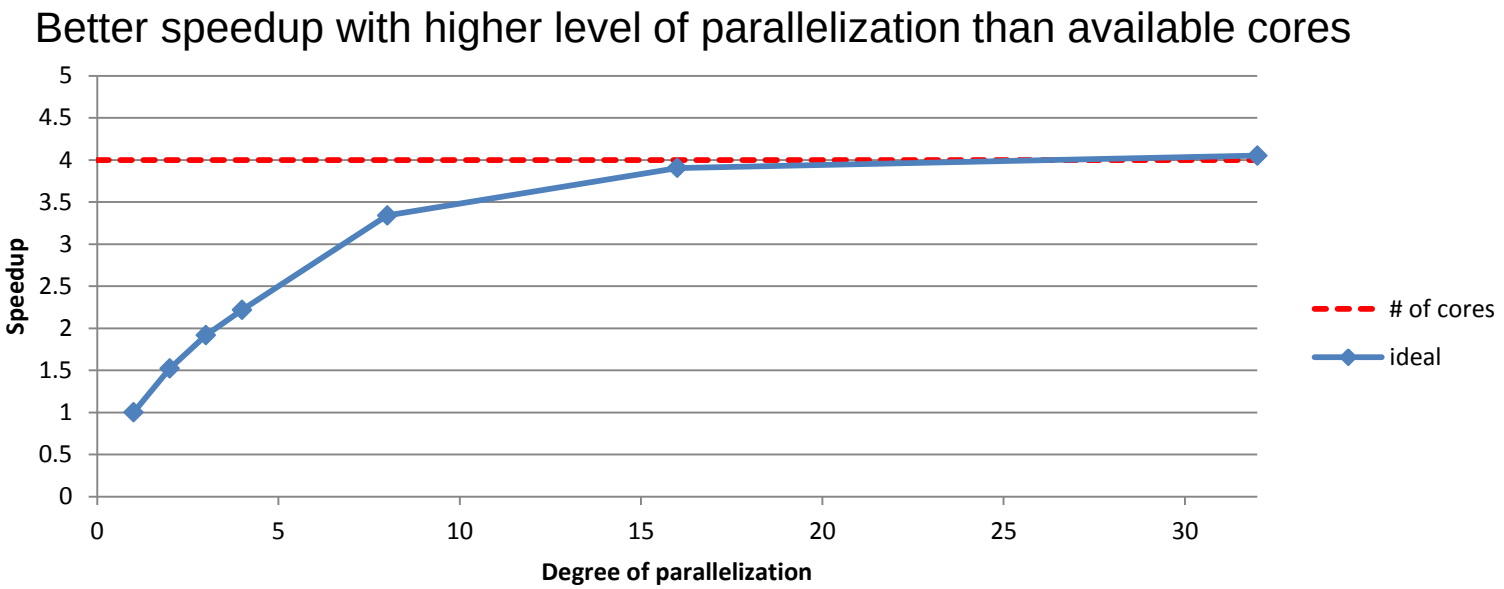


- Trivial to parallelize, high level parallel implementation using .NET PLINQ

```
var slices = from T in temperatures.AsParallel().AsOrdered()
             select singleThreadedSliceCalculationFunc(T);
```

- No manual load balancing or thread/task pooling
- Observed speedup in multithreading up to around 4

Phase envelope multithreading

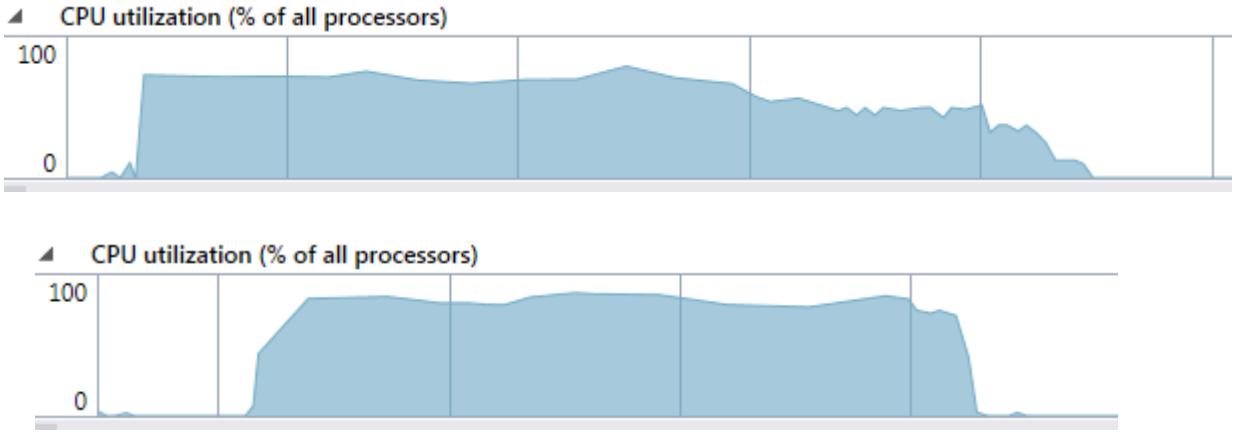


PLINQ degree of Parallelization (DP)

DP = 8

DP = 32

CPU utilization



Summary

- Multiflash native support for multithreaded applications in 2016
- Improved memory footprint
- Good scaling of CPU performance
  - Performance depends on threading implementation by consumer
- How to test CO interface (automated testing)?
  - .NET code to allow validation of interface and values within standard testing framework (e.g. MSTest, Nunit, xUnit.Net)
- Thanks to Jasper van Baten (amsterCHEM) for help and discussions