SWIFT2: High performance software for short-medium term ensemble streamflow forecasting research and operations

Jean-Michel Perraud, Robert Bridgart, James C Bennett and David Robertson

CSIRO Land and Water
Email: jean-michel.perraud@csiro.au

Abstract: The second version of the Short-term Water Information and Forecasting Tools (SWIFT2) is a suite of tools for flood and short-term streamflow forecasting, consisting of a collection of hydrologic model components and utilities. It is currently developed within the Water Information Research and Development Alliance, with a requirement to address both research and operational modelling needs. Catchments are modelled using a traditional approach for semi-distributed models: conceptual subareas and a node-link structure for channel routing. SWIFT2 comprises modules for calibration, model state updating, ensemble runs, output error correction and data assimilation. While applicable to simpler modelling tasks, SWIFT2 is designed from the ground up to tackle computationally intensive problems in probabilistic forecasting research, with sub-daily time steps over decades, and ensemble sizes in the thousands. Typical retrospective ensemble simulation outputs can reach hundreds of gigabytes, and projected runtimes can reach days if not parallelized. SWIFT2 simulations are safe for in-memory copy and parallel executions, can make effective use of multi-core personal computers or high-performance compute clusters to address computationally demanding research or operational requirements. SWIFT2 is written in modern C++ and is platform-agnostic. SWIFT2 is architected to offer several options for user interfaces. Driven by the needs of current users, it is seamlessly accessible from high-level interactive languages including R, Python and MATLAB (Figure 1). To achieve this, the SWIFT API (Application Programming Interface) offers a controlled yet fully fledged access to core modelling features and concepts. The time series handling is performed in a separate library for reuse outside of SWIFT2, using advanced C++ templates, similar to the Boost libraries. Users have access via concise commands in their interactive language of choice to high level modelling concepts, such as simulations, parameterizers, state initializers and objective calculators. SWIFT2 already offers sophisticated features for calibration such as composability of parameter sets, definition of meta-parameters and non-trivial hard constraints on parameter feasibility. This permits the formulation of modelling goals, e.g. calibration, for use in conjunction with state of the art third-party model analysis tools such as metaheuristics frameworks. While the robustness and correctness of SWIFT2 are the primary development goals, runtime performance and parallel scalability are important. Initial measures and usages already show amply adequate operation in calibration mode on multi-core computers, with a very good multi-threaded parallel scalability on larger catchment models. Known performance tuning strategies have been investigated and will further enhance the raw computational performance.

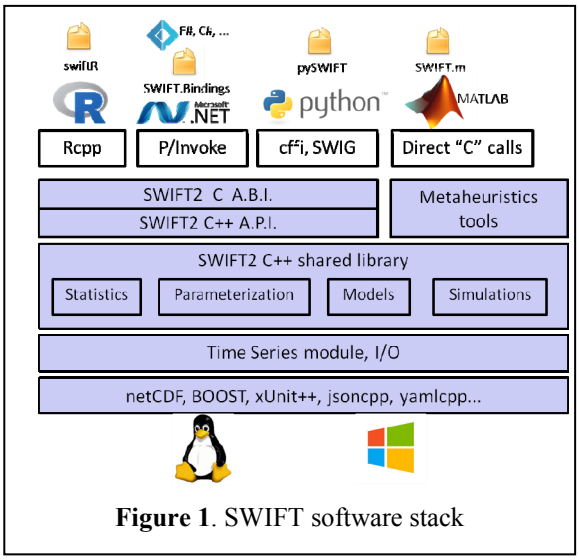

Figure 1. SWIFT software stack

Keywords: *Ensemble streamflow forecasting, high-performance computing, semi-distributed model*

1. BACKGROUND

1.1. Introduction

Ensemble streamflow forecasting at short to medium term lead times (0-10 days) is a burgeoning area of research, promising to produce more accurate forecasts to longer lead times, with formally quantified forecast uncertainties. One major goal of the Water Information Research and Development Alliance (WIRADA) between the CSIRO Land & Water flagship and the Bureau of Meteorology (Bureau) is to provide an operational short to medium term ensemble streamflow forecasting system for Australia. Research output of this alliance consists of new ensemble forecasting methods (e.g. Robertson et al. (2013); Bennett et al. (2014); Li et al. (2015); Shrestha et al. (2015)). These methods are implemented in software to support current research, but with a clearly stated need to transition to an operational use by the Bureau. WIRADA streamflow forecasting software development began with the first implementation of the Short-term Water Information and Forecasting Tools (SWIFT) described in Pagano et al. (2011). SWIFT was primarily designed for continuous deterministic streamflow forecasting. In this paper we describe SWIFT2, a new software that draws on knowledge and methods gained from SWIFT, with enhanced capabilities for short-medium term ensemble streamflow forecasting. SWIFT2 is able to calibrate hydrological and error models used in ensemble streamflow forecasting, generate forecasts retrospectively and in real-time, and be flexible and robust enough to meet the needs of WIRADA researchers and Bureau operational forecasters.

The present paper focuses on the systems design and implementation of SWIFT2. A more user-oriented introduction can be found in Perraud et al. (2015). We very briefly outline the main use cases in the next section, and the resulting system requirements.

1.2. Use cases

Table 1. Modelling use cases

Use case	Description, purpose
Continuous simulation	Daily down to sub-hourly, over a single realization of input time series.
Near real-time forecast	An operational forecaster can hot start a simulation, adjust antecedent moisture, and forecast streamflow using one or more rainfall forecasts.
Calibration	Fitting catchment model parameters to one or more observations, typically one or more streamflow gauges. This may include multiple calibration steps over sub-catchments, staged over successive streamflow gauges. There are cases where the feasibility region of parameter sets includes non trivial constraints, such as Muskingum routing valid over multiple river stretches with different characteristics. Defining virtual parameters in a transformed to make the calibration problem more tractable and robust is also needed.
Ensemble Simulation	A superset of the Continuous simulation use case, adding a dimension to the data in the form of multiple realizations of one or more of the climate input time series.
Retrospective Ensemble Forecasts	At regular intervals in a Continuous simulation, project streamflow at a lead time over one or more forecast of a climate input (typically rainfall forecasts). Storage of ensemble time series and forecasts is required.
Error correction	Blending SWIFT2 model states with observations. This use case is not independent of others above, but one aspect that can be superposed to them.

There are several computational challenges arising from the use cases in Table 1, mostly due to the finer time step resolution or higher number of dimensions compared to what traditional semi-distributed modeling systems tackle. Ensemble forecasts often require a very large volume of data to be processed through hydrological models. Consider a system that generates a thousand-member ensemble streamflow forecasts at an hourly time-step to a lead-time of 10 days, using a semi-distributed conceptual hydrological model with 100 subareas (a fairly typical example). To generate a single forecast with this system requires an input of 2.4×10^7 rainfall data points. The problem becomes particularly acute when generating retrospective forecasts to evaluate the performance of a system – a routine requirement in ensemble forecasting - which usually requires generating and archiving many years' forecasts for a range of catchments.

1.3. Requirements

To address research needs, forecasting software has to be versatile and flexible enough to accommodate new and sometimes complex mathematical method. Primary users of SWIFT2 are using high level languages such as MATLAB, R and Python to do their research, and a seamless access to SWIFT2 is needed. The computational load in terms of runtime and data size, especially for research needs, requires that parallel

execution is possible for SWIFT2 on systems ranging from multi-core personal computers to computational clusters. The rise of "cloud" based infrastructure over the recent years is also a development that may impact on the SWIFT2 system architecture, though not in the immediate future.

Operational streamflow forecasts may be used for flood warnings or other crucial services, so software suitable for operational forecasting has to be extremely reliable. In addition, forecasts have to be reproducible in order to train forecasters and to demonstrate that the system is robust. Finally, the operational forecasting software has to be readily maintainable and upgradeable to enable easy implementation of new forecasting methods. SWIFT2 can be operated from the Flood Early Warning System (FEWS), using input/output adapters to exchange data between the two software systems.

2. SOFTWARE STACK

The previous version of SWIFT was written in Fortran 95. We attempted to evolve that version to a modelling system more in line with state of the art architectural patterns found in environmental modelling framework. While the most recent standards of Fortran (2003 and 2008) include features such as some object orientation useful to such a transition, we found development environments and non-numeric Fortran libraries were lacking to grow and manage the evolution of the system. SWIFT2 is thus a necessary rewrite in C++, albeit with known approaches to include legacy Fortran modules should it be required. To be more precise, SWIFT2 uses selected language features up to the C++11 standard, mostly for features where the syntax is clearer and more concise than older standards of C++.

SWIFT2 comprises several libraries, the main of which is a shared (dynamic) library containing modelling abstractions (Table 2).

Table 2. Main modelling abstractions in SWIFT2

Concept	Description, purpose
Time Step Model	Abstraction for lumped modelling processes, responsible for updating its state variables from time step to time step in the simulation.
Node, Link, Subarea	A Catchment (which is a Time Step Model) comprises a node-link graph (river system) with subareas contributing flows to the links.
Simulation	Aggregation of a Catchment, one or more input or output Time Series, a Time Span and a Time Step
Ensemble Simulation	Aggregation of a Catchment, one or more input or output Time Series of forecasts, a Time Span and a Time Step. We strive to keep the code shared with a simple Simulation, despite the fact of two additional dimensions to the content of the time series, from scalar data points to ensemble of forecast time series
Parameterizer	A high level expression of how to parameterize a Catchment. This includes, but is not limited to, hypercubes, as is usually the case in hydrology. This is used to expose free parameters to an optimization toolbox.
Objective Calculator	Aggregate of a Simulation and one or more statistics. Returns one or more objective scores when requested to evaluate a Parameterizer.
State Initializer	Optional component to a Simulation, a high level expression of how to initialize the states its Catchment. Structurally similar to Parameterizer, and indeed can be wrapped by the latter in cases where initial state conditions can be parameters fitted by an optimization. This feature is important for cases where calibration is performed on short data records, to limit recourse to warmup periods.
Time Series	Time series handling, in memory and input/output. Available stand-alone, independent from other concepts. Time series related components are using C++ template programming to facilitate versatility and reuse.

The components dealing with time series, their representation in memory as well as input/output, is a library distinct from SWIFT2 itself. We did search for existing time series libraries in C++, but did not find instances suitable for our needs without significant downsides (dependencies on particular toolsets, financial modeling concepts, etc.). By writing an independent time series library, other software applications, including some that the authors of this paper wrote, can use the same infrastructure for the management of time series without introducing an unnecessary dependency on SWIFT2 modelling concepts. The preferred time series storage on disk is done with a new convention built for netCDF (Rew et al. 2006). There may be an opportunity to release this time series library publicly with an open source license. Figure 2 shows the declaration of a template time series structure, where the data points can be double, but also

```
template <typename T = double,
    template <typename> class StP = DefaultStorageFloatingPointPolicy,
    template <typename> class MvP = DefaultMissingFloatingPointPolicy
>
class TTimeSeries
```

Figure 2. Time series using C++ templates

complex structures such as time series, for a consistent codebase across different data types. We used C++ template and metaprogramming techniques (Vandevoorde and Josuttis (2002)) to prevent duplication of code yet have sensible behaviors for various item data types in time series. Another benefit of this approach is that more issues that usually arise at program runtime are caught at compilation time.

Table 2 does not list optimization algorithms as a feature in SWIFT2. This is a deliberate choice, as there are several suitable metaheuristics frameworks in C++ or the other languages from which SWIFT2 is accessible. Figure 3 shows that the modeling abstractions and components from Table 2 are in a SWIFT2 shared library (apart from the Time Series deliberately separate). SWIFT2 models have been calibrated using pre-existing metaheuristics tools (for instance https://github.com/jmp75/metaheuristics).

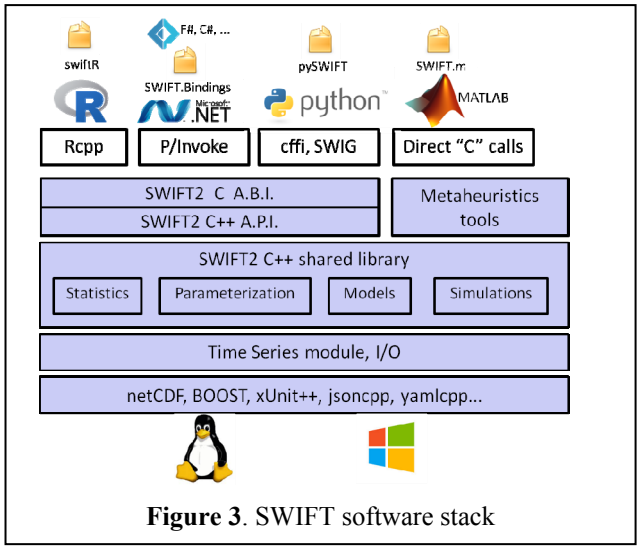

Figure 3. SWIFT software stack

One motivation to use C++ for SWIFT2 is that its subset language C is the most amenable to allow interoperability from the higher level languages used by modelers, and for writing adapters to and for external toolsets such as external optimization modules. While the initial need for a SWIFT2 Application Programming Interface (API) was driven by interactive testing from R, it has proven to be a salient feature of SWIFT2. As of writing, functional access arrangements are accessible from more than 4 high-level languages. Examples of usage of SWIFT2 features from these languages can be found in Perraud et al. (2015). The C API (Figure 4) is designed to offer up to a binary compatibility to users. This provides a very stable programming interface for e.g. swift packages or modules written in R, Matlab, etc. even as SWIFT2 core C++ evolves. Importantly, this minimizes the overhead of memory management: SWIFT2 tracks the use of its main modeling instances via shared pointers (Figure 4) and keeps the heavy lifting to itself, yet making it available to all applications using it at a minimal cost via functions that hide as much complications as possible.

```
#if !defined(SWIFT_USE_OPAQUE_POINTERS)
#define OBJECTIVE_EVALUATOR_PTR  SharedPointer<ObjectiveEvaluator>*
#define PARAMETERIZER_PTR  SharedPointer<Parameterizer>*
#else
#define OBJECTIVE_EVALUATOR_PTR void*
#define PARAMETERIZER_PTR void*
#endif

extern "C" {
    SWIFT_API double EvaluateScoreForParameters(
        OBJECTIVE_EVALUATOR_PTR objectiveEvaluator, PARAMETERIZER_PTR parameterizer);
    // etc.
```

Figure 4. SWIFT C/C++ A.P.I. with opaque pointers

3. COMPUTATIONAL SCALABILITY

While the temporal and spatial resolution of model simulations in SWIFT2 is customizable, it is designed from the start to handle water yield simulations typically at an hourly or finer temporal resolution. Compounding the computational size, primary use cases involve ensemble simulations or ensemble forecast simulations, with an ensemble size in the thousands. This is at least an order or magnitude more than we tackled a few years ago, and to our knowledge there is no pre-existing catchment modelling software tools that could be built upon to scale up to our problem size.

This section reports an initial assessment of runtime performance. While runtimes and scalability have already been more than adequate to perform full scale whole-catchment calibration, we emphasize that the figures are only a snapshot mid-2015 that will be used as a baseline. We expect significant speedup improvement over the coming year. We have prioritised software correctness and robustness, though of course design for runtime performance was taken into account upfront when achievable. Further performance tuning strategies are known, and will be enacted upon when needed.

3.1. Method

This section reports measures done on a tree-like subarea-link-node models for various stream orders, with hourly inputs. While generated, these test case are structurally very close to real models.

We present measures for a system with stream order 4 (30 subareas) in Figure 7. Each point is the median of 10 replicate test runs to reduce measurement uncertainty.

Figure 7b includes on the ordinate axis is a performance measure, where 1 is assigned to the best performing test case.

$$S = R / (L\ N) \quad (1)$$

Where R is the test case total runtime length (s), L the simulation length (years) and N the number of subareas, equal to the number of links. S is thus the average processor time required to run each subarea/link in a model, for each simulation year.

Let S_0 be the lowest S. We define the performance for each test case j as:

$$P_j = S_0 / S_j \quad (2)$$

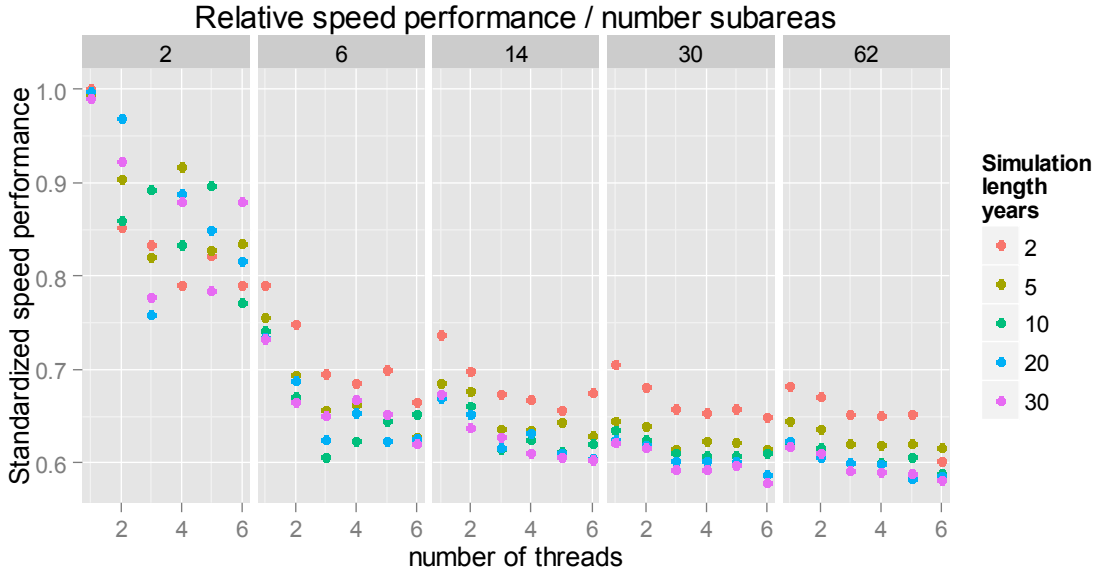

Figure 5. Scalability of runtime performance with respect to system size, simulation length and degree of parallelism. Each point is the median of 10 replicate test runs to reduce measurement uncertainty.

3.2. Observations

Intuitively (perhaps), the fastest performing test case is for the smallest problem size with respect to the three factors simulation length, number of subareas and level of parallelism (Figure 5). As the number of subareas increases from 2 to 62, we observe a drop in performance, but the performance score tends to plateau and is largely within 60 to 70 percent for catchment models with more than 14 subareas. For each case with 14, 30 and 62 subareas the data points are not markedly different.

Looking at the cases with 30 and 62 subareas, we notice that for a given number of threads, we observe a drop of performance for longer time spans of the simulation, however the decreases almost stops beyond 10 years of simulation time span. The second pattern, and a pleasant one, is that the drop in the performance score for higher number of parallelism (threads) is small, and getting smaller for larger simulations.

Looking at each case for smaller catchments (2 and 6 subareas), the pattern of evolutions with each factor is less clear, in spite of a measurement protocol that uses 10 replicates to reduce noise. While not a surprise, we do not have yet an explanation as to the behavior. We do observe that for a 2 subareas system, the simulation length has no impact on the performance for a single thread run, whereas moving to multiple does decrease performance (as expected) but without a clear pattern for how simulation length impacts performance, for a given number of threads.

We measure that a model with 30 subareas requires ~100 milliseconds for one simulation year, at an hourly time step, using AWBM and linear Muskingum routing scheme. In practice, given the full support in SWIFT2 for parallel model computations, this performance is more than adequate for the calibration needs we have had so far and the ones we foresee. Projected runtimes for ensemble forecasts are also expected to be

adequate, and if not the bottleneck is more likely to be input and output of data on disk than raw computational speed.

We intuit that the main approach to improve scalability with respect to numbers of subareas and simulation length is a vectorization of the code. One way to describe it is that we swap the order of the time and geolocation dimensions in how the time series data is stored in memory. This change is provisioned for, but not a priority. Note also this change may be at the detriment of scalability with multi-threading, though an overall improvement would be expected still.

4. POTENTIAL AND CONCLUSION

SWIFT2 is used by project members for core research purposes in calibration and simulation mode. Retrospective ensemble forecasting and error correction modeling capabilities are currently being ported from experimental R code to SWIFT2 in C++, for wider use and transition to the Bureau. While there will be some changes in the technology stack and ongoing incremental improvement to these features, we do not anticipate a need for major changes. The Bureau of Meteorology is implementing an interoperability layer to access SWIFT2 from the Python language. There is further potential for reuse of SWIFT2 for modelling needs that are beyond the scope of the current project, though current efforts are aimed at tying together and operationalizing the features for use by team members for research, then operational short term streamflow forecasting purpose.

Computational requirements were a strong incentive, though not exclusive, to choose C++ as a language for the core of SWIFT2. Most users would not choose C++ as a language to express and manage their modeling workflows, however. The design of the SWIFT2 API has already been used to offer more convenient access to features from several high-level, interactive languages. One benefit is that the growing set of tools in these higher level languages and platforms fostering reproducible research can be used to manage modelling done with SWIFT2. This calls for further enhancements to the SWIFT2 API to interface with data provenance management systems, especially in a context of transition to operations.

ACKNOWLEDGEMENT

This work is carried out in the CSIRO Water for Healthy Country National Research Flagship and is supported by the Water Information Research and Development Alliance between CSIRO and the Australian Bureau of Meteorology.

REFERENCES

Bennett, J. C., Robertson, D. E., Shrestha, D. L., Wang, Q. J., Enever, D., Hapuarachchi, P., and Tuteja, N. K. (2014), A System for Continuous Hydrological Ensemble Forecasting (SCHEF) to lead times of 9 days, *Journal of Hydrology*, 519, 2832-2846.

Li, M., Wang, Q. J., Bennett, J. C., and Robertson, D. E. (2015), A strategy to overcome adverse effects of autoregressive updating of streamflow forecasts, *Hydrol. Earth Syst. Sci.*, 19(1), 1-15.

Pagano T.C., Hapuarachchi H.A.P., Shrestha D.L., Ward P., Anticev J. and Wang Q.J. (2012) Hydrologic modelling with Short‐term Water Information Forecasting Tools (SWIFT) to support real-time short-term streamflow forecasting, WIRADA: Science Symposium Proceedings, Melbourne, Australia, 1–5 August 2011. CSIRO: Water for a Healthy Country National Research Flagship, (2012), pp 99-105.

Perraud, J.-M., Bennett, J. C., Robertson, D. E. and Ward, P. (2014), Model Configuration And Data Management In The Short-Term Water Information Forecasting Tools, 11th International Conference on Hydroinformatics, New York, USA, International Association for Hydrologic Sciences.

Perraud, J.-M., , Bennett, J. C., Bridgart, R.J. and Robertson, D. E. (2015). SWIFT2: Advanced Software for Continuous Ensemble Short-term Streamflow Forecasting. 36th Hydrology and Water Resources Symposium, Hobart, Australia 7-10 Dec 2015

Rew, R. K., Hartnett E. J. and Caron J., (2006) NetCDF-4: Software Implementing an Enhanced Data Model for the Geosciences, 22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, AMS (2006).

Robertson, D. E., Shrestha, D. L., and Wang, Q. J. (2013), Post-processing rainfall forecasts from numerical weather prediction models for short-term streamflow forecasting, *Hydrol. Earth Syst. Sci.*, 17, 3587–3603.

Perraud J-M et al, SWIFT2: High performance software for short-medium term ensemble streamflow forecasting research and operations

Robertson, D. E., Wang, Q. J., Bennett, J. C., Shrestha, D. L. , Li, M., Song, Y., Perraud, J.-M., Enever, D., Hapuarachchi, H. A. P. and Tuteja, N. K., (2015), Scientific foundations for a continuous ensemble flood and short-term forecasting service, 36th Hydrology and Water Resources Symposium, Hobart, Tasmania, Engineers Australia.

Shrestha, D. L., Robertson, D. E., Bennett, J. C., and Wang, Q. J. (2015), Improving Precipitation Forecasts by Generating Ensembles through Postprocessing, *Monthly Weather Review*, published online, doi: 10.1175/mwr-d-14-00329.1.

Vandevoorde, D. and Josuttis, N.M. (2002), C++ Templates: The Complete Guide, Addison-Wesley Professional, November 2002, ISBN: 0201734842, 552 p.