International Journal of Population Data Science

Journal Website: www.ijpds.org

Swiftlink: a distributed, parallelised and scriptable linkage engine architecture for the 21st century

Farrow, J^{1,2}

¹Farrow Norris

²SANT DataLink

Introduction

Many currently extant generic tools for performing bulk linkage were developed last century and while they have accreted features to improve their utility they have failed to fully exploit modern computing architectures. Custom highly-parallelised and distributed solutions exist but there is a need for a modern generalised linkage toolkit.

Objectives and Approach

The aim of this project has been to design, from the ground up, a modern flexible, i.e. scriptable, linkage suite which also takes advantage of contemporary computing architectures to address issues such as distribution of computation, parallelisation of computation and cloud-based computation. Rather than being a monolithic linkage tool or a programming suite, a domain specific language has been developed to specifically describe linkage tasks. Linkage tasks written in this language are then 'executed' or 'compiled and run' to perform pair-wise calculations on data elements. Since linkage tasks are generally bespoke, scriptability has been an important consideration.

Results

Developing a domain specific linkage language has enabled problem specification to be more descriptive and flexible than a monolithic linkage application. The shift in focus from a programming language to a linkage language has enabled a corresponding shift in focus to linkage-related effort (such as blocking and comparison strategies) away from distracting 'glue' code which relates not to linkage tasks under consideration but rather to the distracting bookkeeping aspects of programme execution. The same linkage task may be compiled against different back ends and languages, e.g. FEBRL (python), swift, Amazon lambda (go). The architecture has enabled otherwise idle computing resource to be utilised as well as cloud-based computing facilities for increased throughput and performance. The architecture of the linkage system will be shown with examples.

Conclusion/Implications

Contemporary advances in computing sciences can and must be leveraged in modern linkage tools. By providing a custom scriptable linkage language, tasks may be more clearly specified in a manner more flexible than monolithic linkage applications and by uncoupling linkage specification from execution, linkage may be performed optimally across multiple machines and resources.