Document No: WG21 N4308
Date: 2014-11-12
References: ISO/IEC PDTS 19570
Reply To: Barry Hedquist <beh@peren.com>
INCITS/PL22.16 IR

National Body Comments

ISO/IEC PDTS 19570

Technical Specification: C++ Extensions for Parallelism

Attached is WG21 N4308, National Body Comments for ISO/IEC PDTS 19570, Technical Specification – C++ Extensions for Parallelism.

Document numbers referenced in the ballot comments are WG21 documents unless otherwise stated.

NB Comments: PDTS 19750, C++ Extensions for Parallelism

Date:2014-11-11	Document: SC22 / WG21 N4308	Project: 19750

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
DE 1				ge	The German NB is asking to consider N4167 before finishing the TS. We also belief that it is worthwhile to discuss in how far the goals of transform_reduce can be achieved by a lazy evaluation of transform. More generally, we wish a discussion whether more functional implementations can avoid the need for many new functions which combine existing ones.		

JP 1	4	2.1	Example	ed	Typo for variable names in the example.
					(not 'vec', but 'v')
						(original)
						std::sort(vec.begin(), vec.end());
						
						(correct)
						std::sort(v.begin(), v.end());	

US 1		2.1		ge	The stated scope for execution policy is a good starting point but is insufficient for expressing parallel performance considerations. An execution policy should indicate how parallel execution is supported.	An object of an execution policy type indicates to an algorithm how parallel execution is supported and expresses the requirements on the element access functions.	
JP 2		2.2		te	vector_execution_policy should be added. Although parallel_execution_policy and parallel_vector_execution_policy are defined, vector only execution policy is not. It is important to control the number of threads, especially for the server side programming. It allows the execution of element access functions to be interleaved on a single thread.	Add vector_execution_policy to header <experimental/execution_policy> and new subclause for it.	
CH 1		2.3	3	Te	Implementations should be allowed to add experimental execution policies, especially for executors	Replace paragraph 3 by the original wording. The effect of specializing its execution policy for a type which is not defined by library is unspecified (Note: This provision reserves the privilege of creating non-standard execution policies to the library implementation.	
US 2		3.1	2	te	Requiring *all* exceptions thrown during invocations of element access functions to be captured is likely to impose severe scalability limitations.	The initial list of uncaught exceptions thrown during the invocations of element access functions shall be contained in the exception_list. The size of this list may be bounded.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
JP 3	11	4.1.2	Paragraph 4	te	In the paragraph, only the conditions where standard library function is vectorizaion-unsafe are specified. But it is not clear. The specification should explicitly state which standard library functions are vectorizaion-safe.	Add a list of vectorization-safe standard library functions.	
US 3		4.1.2	3	te	The specification should make it clearer that element access functions and the like do not visibly interrupt other user-visible threads. For example, an element access function may not run between a system call that sets errno and the examination of errno on a user-created thread.	This may require the specification to introduce some additional terminology.	
US 4		4.4.2, 4.4.3		ge	In addition, reduce and exclusive scan algorithms need 'function' versions similar to for_each. Each invocation of an element function shall produce a contribution for the reduce or exclusive scan. This is an important pattern for parallel algorithms.	template< class InputIterator, class T, class BinaryOperation, class Function> T reduce(InputIterator first, InputIterator last, T init, BinaryOperation binary_op, Function f); template<class InputIterator, class OutputIterator, class T, class BinaryOperation, class Function> OutputIterator exclusive_scan(InputIterator first, InputIterator last, OutputIterator result, T init, BinaryOperation binary_op, Function f);	
JP 4		4.4.2-4.4.4		te	In each subclause, prototypes for reduce, exclusive_scan and inclusive_scan don't have an argument for execution policy.	Add an argument for the execution policy as suggested below. (original) template<class InputIterator> typename iterator_traits<InputIterator>::value_type reduce(InputIterator first, InputIterator last); (suggested) template<class ExecutionPolicy, class InputIterator> typename iterator_traits<InputIterator>::value_type reduce(ExecutionPolicy&& exec, InputIterator first, InputIterator last);	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

NB Comments: PDTS 19750, C++ Extensions for Parallelism

Date:2014-11-11	Document: SC22 / WG21 N4308	Project: 19750

MB/ NC[1]	Line number	Clause/ Subclause	Paragraph/ Figure/Table	Type of comment[2]	Comments	Proposed change	Observations of the secretariat

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by **)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial