Potrzebne statystyki awarii na poziomie systemu

paź 6, 2025

Teraz, gdy w VirtuosoNext wprowadzono obsługę odzyskiwania po awarii, zastanawialiśmy się, jak szeroki mógłby być jej zasięg w rzeczywistych systemach. Problem polega na tym, że dane dotyczące przyczyn awarii są albo traktowane jako poufne, albo wąsko koncentrują się na określonych elementach (np. niezawodności sprzętu). Nie możemy znaleźć danych statystycznych dotyczących tych awarii na poziomie systemu. Czy znasz takie dane?
W rzeczywistym systemie mamy warstwy i pewne założenia. Po pierwsze, dzisiejszy sprzęt można uznać za wysoce niezawodny. Oczywiście zakłada się, że przestrzegano zasad projektowania. Jeśli sprzęt ulegnie awarii, najczęściej będzie to spowodowane awariami wprowadzonymi z zewnątrz (przesunięcia bitów, skoki napięcia zasilania itp., problemy z wejściem/wyjściem itp.). Po drugie, oprogramowanie może być poprawne (np. gdy jest formalnie opracowane i sprawdzone), ale prawdopodobnie nadal będzie zawierało błędy resztkowe. Mogą one wynikać z niekompletnych specyfikacji, niestabilności numerycznej, błędów kompilatora, naruszeń dostępu do pamięci itp. Aby uprościć sprawę, musimy również założyć, że sprzęt zapewnia pewne wsparcie w wykrywaniu takich awarii. Układy zarządzania pamięcią mogą wykrywać naruszenia dostępu do pamięci, niedozwolone instrukcje i błędy danych, które mogą generować przerwanie wyjątku, a na poziomie bardziej szczegółowym przekroczenia limitu czasu mogą sygnalizować, że cała jednostka nie odpowiada. Wszystko inne może wymagać redundancji w architekturze.
Powyższe wsparcie ma na celu zapewnienie ciągłości działania aplikacji wbudowanych czasu rzeczywistego, nawet w przypadku wystąpienia powyższych błędów. Środowisko programistyczne wspomaga precyzyjny podział przestrzeni i czasu, ale pozwala również na definiowanie automatycznych działań „czyszczenia i odzyskiwania”. Generatory kodu można rozszerzyć o automatyczne generowanie redundancji czasowej i przestrzennej (ponieważ VirtuosoNext jest transparentny dla procesorów wielordzeniowych).
Wiele funkcji takiego wsparcia odzyskiwania po awarii można oczywiście zaprogramować ręcznie, ale idealnym rozwiązaniem jest automatyzacja. To ostatnie powinno opierać się na analizie kompromisów. Należy pamiętać, że obecnie praktyka jest często oparta na precyzyjnym podejściu. W przypadku wystąpienia błędu cała aplikacja, a nawet cały system przetwarzania, zostaje ponownie uruchomiona. Nawet jeśli takie zdarzenie ma niskie prawdopodobieństwo wystąpienia, w wielu przypadkach może mieć katastrofalne skutki. Czas rozruchu może być stosunkowo krótki w przypadku małych programów (kod musi zostać odczytany np. z pamięci flash, a system ponownie zainicjowany), ale jeśli ograniczenia czasowe są zbyt krótkie (np. mikrosekundy lub milisekundy), a kod jest stosunkowo duży, nie jest to realna opcja. Dlatego system powinien uniemożliwić restart jako ostatnią dostępną opcję.
Aby zapewnić takie wsparcie w sensowny (i ekonomiczny) sposób, musimy dowiedzieć się więcej o resztkowym prawdopodobieństwie awarii i błędów w rzeczywistym (wbudowanym) systemie. Nie możemy znaleźć danych statystycznych dotyczących tych awarii na poziomie systemowym. Czy znasz takie dane? Zdajemy sobie sprawę, że może to nie być trywialne, ale Twoja pomoc będzie bardzo doceniona.